Applied Simulated Annealing Algorithm in Resource Allocation for Virtual Service Based on Heterogeneous Shared Hosting Platforms

Nguyen Minh Nhut Pham[1,*], Bao Hung Hoang[1]
[1]Vietnam Korea Friendship Information Technology College
Danang, Vietnam
[*]Email: minhnhutvh [AT] gmail.com

Van Son Le[2]
[2]Departement of Informations, University of Education, Danang University
Danang, Vietnam

Abstract—**Providing resource for virtual services in cloud computing which requires saving the resource and minimizing the amount of energy consumption is critical. In this study, we propose the resource model and linear programming formulation for multi-dimensional resource allocation problem. Based on the simulated annealing algorithm, RASA algorithms were designed to solve and evaluate through Cloudsim simulation tool compared with FirstFit Decreasing (FFD) algorithm. The parameters include the number of physical machines being used and the amount of energy consumption. The experimental results show that the proposed RASA algorithm yields a better performance than FFD algorithm.**

Keywords-Resource Allocation; Cloud Computing; Virtual machine; Simulated Annealing

I. INTRODUCTION

Technology development and the applicability of cloud computing foster the growth and explosion of data centers which related to the high-demand of power using up and threats to the environment. The optimal resource allocation for virtual services with quality supply and the minimal energy consumption in these areas have been interested in by many researchers and are the key points to study in this paper as well.

Eugen Feller *et al.* [7] showed a resource allocation model to provide the number of physical machines to virtual services and used an optimization algorithm "ants" to estimate. This previous study's simulation results proved that the energy consumption of the system is reduced if the number of used physical machines is diminished. However, instead of considering the homogeneous platform and experiment on the simulated data, we will explore the heterogeneous physical machines environment such as resources configuration of the physical machines are not the same, then attempt the real given data [1, 2].

Mark Stillwell *et al* [14] established the resource allocation problem as a linear programming obstacle and used a *FFD* algorithm to assess. However, Thomas Setser [15] demonstrated that, this algorithm tend to wasting capabilities. According to some reviews [4, 6, 9, 13] pointed out some practices of system's resources allocation with the minimal energy consumption and just focused on power utilization on the physical machines' CPU. Furthermore, another researches [10, 12] believed that this burning is not only upon CPU but also over other appliances such as hard disk, bandwidth, RAM, ect.

Therefore, we propose the *RASA* algorithms based on Simulated Annealing method to solve this dilemma. We analyze the multi-dimensional resource allocation issue (physical resources) for virtual service from a heterogeneous shared hosting platform. The aim is lessen the energy use from all of sources through curtailing the used physical machines. The key contributions of the paper are as follows:

- Building the approaches of capability demand, resource allocation, and energy consumption for virtual service from the heterogeneous shared hosting platforms with condition that each virtual service will be a single virtual machine.
- Modeling a resource allocation for virtual service from the heterogeneous shared hosting platforms as a linear programming problem;
- Developing the *RASA* algorithms which are based on the Simulated Annealing method and using Cloudsim tool [3] to solve. Comparison of energy consumption, the number of used physical machines and the execution time between *RASA* and *FFD* algorithm through real data.

Additionally, the authors divide this study into four parts: Section 2 presents a mathematical model as linear programming case, an energy consumption model. Section 3 proposes the RASA algorithms to solve the problem. Section 4 follows with experimental results and comparisons in various scenarios. Section 5 is conclusion and suggests the future work.

II. RESOURCE ALLOCATION FOR VIRTUAL SERVICE MODEL

A. *Resource*: *the needs and approach*

A heterogeneous shared platform includes a cluster of physical machines has different resource configuration and is

interconnected by a high-speed network device is deployed for sharing resources to virtual services is carried out in this paper.

For each type of ability at an objective engine might have one or more distinct elements such as one or more separate real CPU, one or more single substantial memory, ect. and aggregate one. Thus, the capabilities of a corporeal appliance are represented by an ordered pair of multi-dimensional resource vectors. The elementary resource vector expresses capacity of a single aspect while the aggregate resource vector executes sum of resource capacity counting all factors.

Similarly, the virtual machines are showed by virtual single elements and virtual aggregate elements; the resource needs of virtual service are performed by an ordered pair of multi-dimensional resource needs vectors, including elementary one and aggregate one. In fact, each virtual service has two kinds of resource needs: *rigid needs* and *fluid needs* [14]. A *rigid needs* acts as a specific fraction of required resource. The service cannot benefit from a larger fraction and operate with a smaller one as well. A *fluid needs* specifies the additional fraction of a resource that the service could use. The service cannot gain from a bigger portion, but can run with a minor one at lower cost.

Hence, the rigid needs of resource type k of service i are given by an ordered vector pair (r_{ik}^e, r_{ik}^a), that represents the resource needs to run the service at the acceptable service level. With r_{ik}^e , r_{ik}^a to denote the elementary rigid needs and aggregate rigid needs of service i for resource type k , respectively. If this resource needs cannot be satisfied, then resource allocation would fail. The fluid needs of resource type k of service i are presented by a second ordered vector pair (f_{ik}^e, f_{ik}^a) which displays the additional resources demanded to run the service at the maximum level of performance. In particular, f_{ik}^e , f_{ik}^a stand for the elementary fluid needs and aggregate fluid needs of service i for resource type k , respectively. So, the fluid needs is defined by the multiplication of the fluid needs of resource and the additional factor of service which is called a *constrained fluid needs*.

The utilizations of all resources corresponding to fluid needs are linearly correlated. For example, if the service is only provided a half of the CPU resource needs, the possibility is that it only uses half of the bandwidth I / O compared with the resource needs. This is consistent with the reality because when the CPU resource needs cut down, this leads to the reduction of other resources consumption (in this case is the bandwidth I /O). For simplicity, the additional factor of all fluid needs can show the same value and its value is between 0 and 1, with 0 coinciding with the case in which the service is not provided the fluid resource, whereas with 1 conforming to the service which executes at maximum resource allocation. Therefore, resource needs of resource type k of virtual service i on physical machine j with additional factor Q_{ij} are given by an ordered vector pair $(r_{ik}^e + Q_{ij} f_{ik}^e, r_{ik}^a + Q_{ij} f_{ik}^a)$.

Similarly [2], we assume that elementary resource needs of a virtual service does not exceed the elementary resource capacity of a physical machine. And, providing resource also fail if the aggregate resource needs of virtual services exceeds the capacity of physical machines aggregate resources.

Figure 1 illustrates a simple illustration of the resources and resource needs model with two physical machines and one virtual service. Specifically, the physical machine A comprises 4 CPU cores such as 4 CPU single resource elements and 1 RAM for instance 1 RAM single resource element. In this way, the CPU resource vector pair is (1.0, 4.0) and RAM resource vector pair is (2.0, 2.0). Physical machine B comprises 2 cores and 1 RAM. So, the CPU resource vector pair is (1.0, 2.0) and RAM resource vector pair is (1.0, 1.0). For virtual service, the CPU resource needs vector pair of virtual service is $(0.5 + Q_{ij} \times 0.5, 2.0 + Q_{ij} \times 1.0)$, and the RAM resource needs vector pair is $(1.0 + Q_{ij} \times 0.0, 1.0 + Q_{ij} \times 0.0)$. If Q_{ij} is assumed Q_{ij} =1 then the physical machine A fully satisfied in resource allocation to virtual service. Otherwise, Q_{ij} is assumed Q_{ij} =0.5, the physical machine B is not only fully satisfied in resource allocation but also the resource cost is decreased.

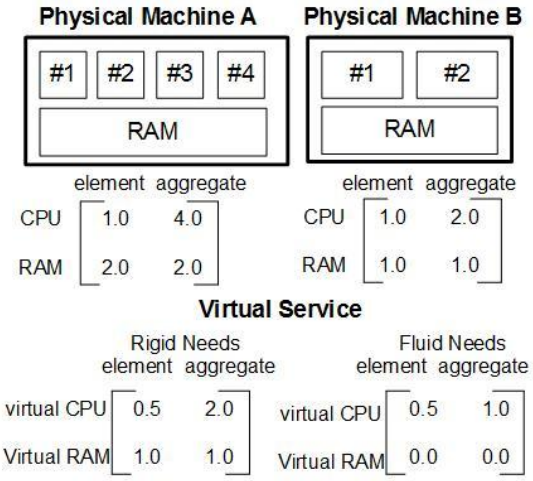

Figure 1. Example of problem instance with resource of two physical machine and resource needs of one virtual service.

B. Objective and constraints

Assuming that each virtual service consists of a single virtual machine which has a fixed resource needs. A static multi-dimension resource allocation problem for virtual service from the heterogeneous shared hosting platforms (HMDRA) is formulated by a Mixed integer linear programming problem (MILP) as follows:

Let S be virtual services, indexed by $i = 1, \ldots, n$; $S > 0$, Y be physical machines having the different resource configuration, recorded by $j = 1, \ldots, m$; $Y > 0$. Each physical machine provides D types of resource, listed by $k = 1, \ldots, D$. Use r_{ik}^e , r_{ik}^a to denote the elementary rigid needs and aggregate rigid needs of service i for resource type k. f_{ik}^e, f_{ik}^a stand for the elementary fluid needs and aggregate fluid needs of service i for resource type k. C_{jk}^e, C_{jk}^a imply the elementary resource and aggregate resource of physical machine j for resource type k, and Q_{ij} intends the additional factor of service i on physical machine j. A binary variable x_{ij}

that is equal to 1 if service i is allocated the resource by physical machine j and 0 otherwise. Finally, a binary variable y_j indicates whether the physical machine j is in use or not for resource allocation to S virtual services. Objective is to minimize the number of used physical machines. Given these notations, a HMDRA problem described by a MILP problem are as follows:

$$x_{ij} \in \{0,1\}, Q_{ij} \in [0,1], \quad \forall i,j \qquad (1)$$

$$\sum_j x_{ij} = 1, \quad \forall i \qquad (2)$$

$$y_j \geq x_{ij}, \quad \forall i,j \qquad (3)$$

$$\left(r_{ik}^e + Q_{ij} f_{ik}^e\right) x_{ij} \leq C_{jk}^e, \quad \forall i,j,k \qquad (4)$$

$$\sum_i (r_{ik}^a + Q_{ij} f_{ik}^a) x_{ij} \leq C_{jk}^a, \quad \forall j,k \qquad (5)$$

$$\text{and, object is } \quad min \sum_j y_j \qquad (6)$$

Constraint (1) defines the domain of the variables. Constraint (2) determines the state that resources for service i are provided by exactly one physical machine j. Constraint (3) specifies the state which a physical machine j is used, if it grants resource to at least one virtual machine. Constraint (4) states that the elementary resource of the physical machine j does not overcome, and constraint (5) disposes that the aggregate resource of the physical machine j does not overcome. Finally, constraint (6) is the optimization objective to scale down the number of used physical machines.

C. Energy consumption model

In order to estimate the energy consumption of the physical machines, we choose approximately the power consumption at a physical machine j as a linear function $P_j(u)$ by the formula (7).

$$P(u_j) = (P_{max} - P_{idle}) \times u_j + P_{idle}, \quad \forall j \qquad (7)$$

Among them, P_{max} and P_{idle} are the power of physical machine j in the maximum used utilities state and idle state, respectively. u_j is the total used utilities of all resources on the physical machines j, $u_j \in [0,1]$, and it is calculated by a formula (8).

$$u_j = \sum_{k=1}^{d} \frac{U_{jk}}{C_{jk}}, \quad \forall j \qquad (8)$$

In there, U_{jk} is a resource k of a physical machine j that it is allocated for virtual services, and C_{jk} is the capacity of resource k on a physical machine j. The physical machines are not used, it will be shut down. Therefore, the energy consumption of the Y physical machines in the period t is set as a formula (9).

$$E(t) = \begin{cases} t \times \sum_{j=1}^{m} P(u_j), & if \quad u_j \neq 0 \\ 0 & otherwise \end{cases} \qquad (9)$$

The HMDRA is the problem of combinatorial optimization, NP- complete. Currently, the methods usually are used to clarify combinatorial optimization obstacles, included the heuristic search to find a good enough solution [5]; the local search to find local optimal quick fix [11]; the approximate search through natural simulation algorithms [8]: SA, ACO,... In the following, we propose an improved algorithm based on the Simulated Annealing algorithm [16] for working out.

III. SIMULATED ANNEALING ALGORITHM FOR RESOURCE ALLOCATION PROBLEM

The Simulated annealing algorithm is a probabilistic search method to find global optimal solution for the optimization problems. It reproduces regular metallurgical process in which a steel crystal is heated to a high temperature and is cooled slowly to one in a hardest state (optimal). If the cooling is slow enough, a final result is a steel crystal with a best structure (global optimization). In contrast, a steel crystal will be crystallized in a stable state (local optimization).

The algorithm is used a temperature global variable T. Initially, T is a very high value and it is decreased to the lower value after each of iteration. In specific iterativeness, the cost function is considered for two solutions: the current solution and the neighboured solution. If the neighboured solution is better than the current solution, it will be selected. In contrast, the neighboured one can still be accepted in the hope of escaping local extreme for searching global extreme with a probability which is depends on the cost function value between the current one, so far best explanation and parameter value T.

A. Configuration of resource allocation problem

The result of providing resources from the Y physical machines to the S virtual services is represented by a one-dimensional integer array, $allocation$. Exclusively, the array index is the virtual service and the array element value is the index of physical machine. For example, with 10 virtual services and 3 virtual machines, *allocation*[i] = {1, 3, 1, 3, 2, 1, 3, 2, 1, 2} means that services 1, 3 , 6 and 9 are contributed resource from a physical machine 1; the services 5, 8 and 10 are arranged resource from a physical machine 2; the services 2, 4 and 7 are implemented resource from a physical machine 3. So, the number of used physical machines are 3.

B. Cost function

For Simulated Annealing algorithm, for any iteration at a value of temperature T, current solution is compared with the so far best one to choose a better result which will be a good clarification for the following iteration. That selection is based on the cost function value, f_{cost}. Because the objective of the problem is the minimum number of used physical machine, ie, the minimum provided resources, the cost function is defined as follows:

Let F_{jk} be the total resources of the physical machine j which has provided S_i virtual services. Based on the formula (5), F_{jk} is calculated by the formula (10).

$$F_{jk} = \sum_{i \in j} (r_{ik}^a + Q_{ij} f_{ik}^a), \quad \forall j,k \qquad (10)$$

In consequence, the average resource of the Y_j physical that added the S_i virtual services on all types of resources k are calculated by formula (11).

$$F_{HMDRAP} = \frac{1}{m}\sum_{j=1}^{m}\sum_{k=1}^{d} F_{jk} \quad (11)$$

Notably, m is the number of used physical machines and the cost function is calculated by formula (12)

$$f_{cost} = \frac{1}{F_{HMDRAP}} \quad (12)$$

C. Stop condition

The algorithm will stop after looking for a global optimal value or a temperature as the value T_{min}.

In summary, the psuedo code of *RASA* algorithm to solve a HMDRAP problem is shown as a Algorithm 1

Algorithm 1: *RASA*

Input:

- Number of virtual services *S*, type of resources *D*, elementary rigid needs r_{ik}^{e}, aggregate rigid needs r_{ik}^{a}, elementary fluid needs f_{ik}^{e}, aggregate fluid needs f_{ik}^{a}, and additional factor Q_{ij}.

- Number of physical machines *Y*, elementary resource C_{jk}^{e}, aggregate resource C_{jk}^{a}.

Output: list of used physical machines, $AllocBest$.

```
1: double temp ← T0; coldRate ← Const;
2: AllocBest ← initAlloc(Y,S);
3: double bestCost ← CostFuntion(AllocBest);
4: while (temp > Tmin ) do
5:    for i := 1 to numLoop do
6:       AllocCurr ← AllocBest;
7:       currCost ← CostFuntion(AllocCurr);
8:       AllocNeigh ← allocCurrNeigh (Y,S);
9:       neighCost ← CostFuntion(AllocNeigh);
10:      if (neighCost < currCost)
11:         AllocCurr ← AllocNeigh;
12:         currCost ← CostFuntion(AllocCurr);
13:      end if
14:      if(exp((neighCost − currCost)/temp) > random(0,1))
15:         AllocCurr ← AllocNeigh;
16:         currCost ← CostFuntion(AllocCurr);
17:      end if
18:      if (currCost < bestCost)
19:         AllocBest ← AllocCurr;
20:         currCost ← CostFuntion(AllocBest);
21:      end if
22:   end for i := 0 to numLoop
23:   temp ← temp × (1 − coldRate);
24: end While
25: return AllocBest;
```

Firstly, line 1 is used to initialize the value of temperature *T*, and set the value of the cooling rate, *coldRate*. Lines 2 and 3 implement the initializing solutions to resource allocation from the *Y* physical machines for the *S* virtual services, and estimated the cost of the solution by the formula (12). The computing solutions are run by Algorithm 2 or *FFD* algorithm [14]. From line 4 to line 25 are used to carry out iteration step until the temperature is declined to T_{min}. In each iteration, as purpose is to raise the probability of getting the global optimal value, the algorithm implement *numLoop* emphasises (from line 5 to line 22, and *numLoop* is an experimental parameter).

From line 6 to line 9 are used to implement and estimate the cost function value of a current solution, execulte and assessment the cost function value of a neighbor solution. Next, if the cost function value of a neighbor solution is better than the cost function value of a current solution, the current solution will be the neighbor solution.

If not, the current solution is also a neighbor solution with a probability value which depends on the cost function value of a neighbor solution, the cost function value of a current solution and temperature *T* (from 10 to 17 lines, neighbor solutions is executed by a algorithm 3).

After that, the algorithm will compare the cost of current solution to the cost of the so far best solution. If the value of current solution is better (smaller), the best solution will be the current solution and the cost function value is re-estimate (from line 18 to 21).

Finally, a temperature is cut down (command 23), if the current temperature is a T_{min}, the algorithm returns a best resource allocation solution, *AllocBest* (line 25).

Algorithm 2: *InitRandom*

Input:

- Number of virtual services S, type of resources D, elementary rigid needs r_{ik}^{e}, aggregate rigid needs r_{ik}^{a}, elementary fluid needs f_{ik}^{e}, aggregate fluid needs f_{ik}^{a}, and additional factor Q_{ij}.

- Number of physical machines *Y*, elementary resource C_{jk}^{e}, aggregate resource C_{jk}^{a}.

Output: list of used physical machines, $Allocation$.

```
1: for i := 1 to S do
2:    int j ← random(Y);
3:    for k := 1 to D do
4:       Sum^e_ik ← r^e_ik + Q_ij × f^e_ik ;
5:       Sum^a_ik ← r^a_ik + Q_ij × f^a_ik ;
6:       if(C^e_jk ≥ Sum^e_ik and C^a_jk ≥ Sum^a_ik)
7:          C^a_jk ← C^a_jk − Sum^a_ik ;
8:       end if
9:    end for k := 1 to D
10:   Allocation[i] ← j;
11: end for i := 1 to S
12: return Allocation;
```

Algorithm 3: *AllocNeigh*

Input:

- Number of virtual services S, type of resources D, elementary rigid needs r_{ik}^{e}, aggregate rigid needs r_{ik}^{a}, elementary fluid needs f_{ik}^{e}, aggregate fluid needs f_{ik}^{a}, and additional factor Q_{ij}.

- Number of current physical machines Y_{curr}, elementary resource C_{jk}^{e}, aggregate resource C_{jk}^{a}.

Output: list of used physical machines, $Allocation$.

```
for i := 1 to S do
   int rand ← random(Y_curr);
            for j := rand to Y_curr do
      for k := 1 to D do
        Sum^e_ik ← r^e_ik + Q_ij × f^e_ik ;
        Sum^a_ik ← r^a_ik + Q_ij × f^a_ik ;
        if(C^e_jk ≥ Sum^e_ik and C^a_jk ≥ Sum^a_ik)
           C^a_jk ← C^a_jk − Sum^a_ik ;
        end if
      end for k := 1 to D
      Allocation[i] ← j;
           break;
    end for j := rand to Y
end for i := 1 to S
return Allocation;
```

The combination of the *RASA* algorithm and an *InitRandom* or a *FFD*, we get two algorithms: *Rand-RASA* and *FF-RASA*. Particularly, a *Rand-RASA* is used initiatizing solution from a *InitiRandom* algorithm, and a *FF-RASA* is used initiatizing solution from a *FFD* algorithm. Let S are the number of virtual services, Y are the number of physical machines, D are the dimension of resources. The complexity of algorithms is calculated as follows:

- A initiatizing solution: $O(D \times S \times Y)$
- A neighboured solution: $O(D \times S \times Y)$
- The estimating of cost function: $O(D \times Y)$
- The complexity of algorithms: $O(D \times S \times Y) + O(temp \times numLoop \times (O(D \times S \times Y)+O(D \times Y)))$.

If $D, temp, numLoop$ are considered as a constant, those algorithms have the computational complexity of $O(S \times Y)$.

IV. Numerical Result and Evaluation

A. Simulation

For appraisal of algorithms, we used a CloudSim [4] cloud simulation tool. In particular, we inherited the *Vm* class and *Host* class to expand the characteristic of resource needs for virtual machines and the resources of physical machines. Also, we inherited *VMAlloctonPolicy* class to execute the resource allocation policy for virtual machines that is based on the *RASA* algorithms.

The experimental data are used from the actual data which is presented in [1, 2]. Inside, the resource characteristics and Power consumption of the physical machine are shown in Table 1 and the composition of virtual machine resources similar to the virtual machine of *Amazon EC2* cloud which is modified to suit our problem, as on view in Table 2 and Table 3.

The parameters of the RASA algorithms are used: *temp*=1000; *coldRate*=5; T_{min}=0; *numLoop*=100. For each algorithm, we used three metrics: the number of used physical machines, the consumed energy in period t = 24 hours and the execution time. The Using the number of virtual machines in the experimental scenarios are S = 100; 200; 300; 400; 500. A measure of consumed energy is as kWh and execution time is as seconds (s). The algorithms ran on an Intel(R) Core(TM) i5-3235M 2.60 GHz, RAM 4Gb.

TABLE I. Resource Characteristics and Power Consumption of Physical Machine.

Type of Physical Machine	CPU (MHz)	RAM (GB)	BW (GB/s)	Disk (GB)	P_{idle} (kW)	P_{max} (kW)
HP proliant G4	2core x 1860	4	1	20	86	117
HP proliant G5	2core x 2660	4	1	40	93.7	135
IBM Server x3250	4core x 2933	8	1	600	46.1	113
IBM Server x3550	6core x 3067	16	1	800	58.4	222

TABLE II. CPU resource and RAM resource characteristics of Virtual Machine.

Type of Virtual Machine	CPU(MHz)				RAM(GB)			
	f_{cpu}^{e}	r_{cpu}^{e}	Num. of element	Total	f_{ram}^{e}	r_{ram}^{e}	Num. of element	Total
VM1	1000	1500	1	2500	0.4	0.45	1	0.85
VM2	1000	1000	1	2000	1.0	2.75	1	3.75
VM3	500	500	1	1000	0.7	1.0	1	1.7
VM4	250	250	1	500	0.113	0.5	1	0.613

TABLE III. BW resource and DISK resource characteristics of Virtual Machine.

Type of Virtual Machine	BW(GB/s)				DISK(GB)			
	f_{bw}^{e}	r_{bw}^{e}	Num. of element	Total	f_{disk}^{e}	r_{disk}^{e}	Num. of element	Total
VM1	0.2	0.25	1	0.45	0.5	2.0	2	5
VM2	0.1	0.25	1	0.35	2.0	3.0	2	10
VM3	0.1	0.15	1	0.25	2.5	5.0	2	15
VM4	0.05	0.1	1	0.15	5.0	5.0	2	20

B. Result and evaluation

The experimental results are demonstrated in Table 4 and Figure 2

TABLE IV. THE EXPERIMENTAL RESULTS.

Num. of VM	Algorithms	Num. of Used PM	Times (s)	Energy (kWh)	Gain Energy (%)
100	*FFD*	42	0.031	201.284	
	Rand-RASA	40	0.035	200.913	0.184
	FF-RASA	39	0.037	191.194	5.022
200	*FFD*	80	0.078	396.706	
	Rand-RASA	78	0.090	394.422	0.576
	FF-RASA	75	0.088	392.593	1.043
300	*FFD*	122	0.116	597.989	
	Rand-RASA	120	0.125	588.291	1.622
	FF-RASA	111	0.125	584.564	2.282
400	*FFD*	160	0.144	793.411	
	Rand-RASA	158	0.168	790.405	0.379
	FF-RASA	140	0.164	785185	1.041
500	*FFD*	202	0.200	994.694	
	Rand-RASA	197	0.224	997.367	-0.269
	FF-RASA	189	0.218	990.339	0.437

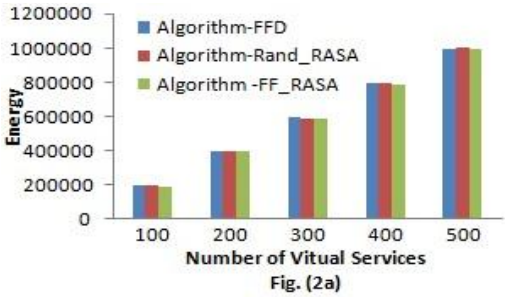

Fig. (2a)

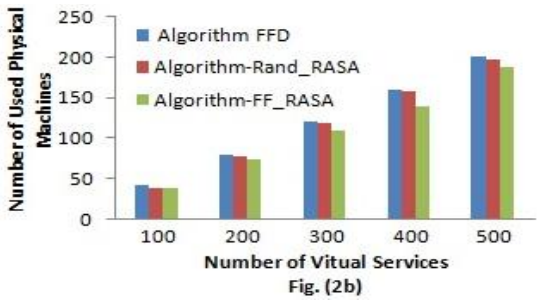

Fig. (2b)

Figure 2. The Graph of used physical machines and consumed energy.

Based on the outcomes in Table 4 and Figure 2, we recognized that a energy is linearly with the number of used physical machines; the value of used physical machines and consumed energy of *FF-RASA* algorithm are the best. But, the execution time of the *RASA* algorithms are larger than *FFD* algorithm. A reason of this is affected by parameter of initial temperature and the number of iterations in RASA algorithms. However, the execution time of the algorithm is small and it can be applied in practice.

V. CONCLUSION AND FUTURE RESEARCH

This paper has dicussed the static multi-dimention resource allocation based on a hetergeneous shared hosting platforms for virtual services with the optimal constraints; each service is considered as a single virtual machine. Standing on the Simulated Annealing method, we proposed the *RASA* algorithms to install, evaluate and compare to the *FFD* algorithm through measures the number of used physical machines, the consumed energy and the execution time . The algorithms are realized by using the CloudSim tool with actual data. The experimental outstanding shows that the proposed RASA algorithm yields a better performance than FFD algorithm. For the future work, the proposed approach will be extended for dynamic resource allocation.

REFERENCES

[1] E. Arianyan, H. Taheri, S. Sharifian, "Novel energy and SLA efficient resource management heuristics for consolidation of virtual machines in cloud data centers". Computers Electrical Engineering 47, 2015, pp. 222 – 240.

[2] A. Beloglazov, R. Buyya, " Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers". Concurr. Comput. : Pract. Exper. 24(13), 2012, pp. 1397–1420.

[3] C. Blum, J. Puchinger, G.R. Raidl, A. Roli, " Hybrid metaheuristics in combinatorial optimization: A survey". Appl. Soft Comput. 11(6), 2011, pp. 4135–4151.

[4] R.N. Calheiros, R.. Ranjan, A. Beloglazov, C.A.F. De Rose, R. Buyya, "Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms". Softw. Pract. Exper. 41(1), 2011, pp. 23–50.

[5] Z. Cao, S. Dong, " Dynamic vm consolidation for energy-aware and sla violation reduction in cloud computing". In: Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2012 13th International Conference on, Dec 2012, pp. 363–369.

[6] F. Farahnakian, A. Ashraf, P. Liljeberg, T. Pahikkala, J. Plosila, I. Porres, H. Tenhunen, "Energy-aware dynamic vm consolidation in cloud data centers using ant colony system". In: Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on, June 2014, pp. 104–111.

[7] E. Feller, L. Rilling, C. Morin, "Energy-aware ant colony based workload placement in clouds". In: Grid Computing (GRID), 2011 12th IEEE/ACM International Conference on, Sept 2011, pp. 26–33.

[8] R. Jansen, P.R. Brenner, "Energy efficient virtual machine allocation in the cloud". In: Green Computing Conference and Workshops (IGCC), 2011 International, July 2011, pp. 1–8.

[9] D. Jia, G. Zheng, M.K. Khan, "An effective memetic differential evolution algorithm based on chaotic local search". Information Sciences 181(15), 2011, pp. 3175 – 3187.

[10] S. Kirkpatrick, "Optimization by simulated annealing: Quantitative studies". Journal of Statistical Physics 34(5), 975–986.

[11] L. Luo, W. Wu, D. Di, F. Zhang, Y. Yan, Y. Mao, "A resource scheduling algorithm of cloud computing based on energy efficient optimization methods". In: Green Computing Conference (IGCC), 2012 International, June 2012, pp. 1–6 .

[12] D.M. Quan, R. Basmadjian, H. Meer, R. Lent, T. Mahmoodi, D. Sannelli, F. Mezza, L. Telesca, C. Dupont, "Energy Efficient Resource Allocation Strategy for Cloud Data Centres", Computer and Information Sciences II: 26th International Symposium on Computer and Information Sciences, chap., 2012, pp. 133–141.

[13] T. Setzer, A. Stage, "Decision support for virtual machine reassignments in enterprise data centers". In: Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP, April 2010, pp. 88–94.

[14] A.M.C. So, "Deterministic approximation algorithms for sphere constrained homogeneous polynomial optimization problems", Mathematical Programming 129(2), 2011, pp. 57–382.

[15] M. Stillwell, F. Vivien, H. Casanova, "Virtual machine resource allocation for service hosting on heterogeneous distributed platforms". In: Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium, IPDPS '12, IEEE Computer Society, Washington, DC, USA 2012, pp. 786–797. (2012).

[16] A. Vigliotti, D.M. Batista, "Energy-efficient virtual machines placement". In: Computer Networks and Distributed Systems (SBRC), 2014 Brazilian Symposium on, May 2014, pp. 1–8.