BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **17**, No 3
Sofia • 2017 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.1515/cait-2017-0029

# Energy Efficient Resource Allocation for Virtual Services Based on Heterogeneous Shared Hosting Platforms in Cloud Computing

*Nguyen Minh Nhut Pham*[1], *Van Son Le*[2], *Ha Huy Cuong Nguyen*[3]

[1]*Vietnam Korea Friendship Information Technology College, Danang, Vietnam*
[2]*Danang University, Danang, Vietnam*
[3]*Quangnam University, Quangnam, Vietnam*
*E-mails: nhutpnm@viethanit.edu.vn levansupham2004@yahoo.com nguyenhahuycuong@gmail.com*

***Abstract*:** *This paper is an extended and updated version, presented at the INDIA 2017 conference. Optimal resource provisioning for virtual services in the Cloud computing is one of the concerns nowadays. For cloud computing service providers, reducing the number of physical machines providing resources for virtual services in cloud computing is one of the efficient ways to reduce the amount of energy consumption, which in turn enhances the performance of data centers. Multi-dimensional resource provisioning on a Heterogeneous Shared Hosting Platforms for virtual services is known as a NP-hard problem. Therefore, it is necessary to apply the metaheuristic algorithms for estimating the outcome of the problem. In this study, we propose the resource allocation problem for reducing the energy consumption. ECRA-SA algorithms were designed to solve it and were evaluated through CloudSim simulation tool compared with FFD algorithm. The experimental results show that the proposed ECRA-SA algorithm yields a better performance than FFD algorithm.*

***Keywords*:** *Resource allocation, simulated annealing, virtual service, cloud computing, energy consumption.*

## 1. Introduction

The development of virtual technology and the applicability of cloud computing have led to the rapid growth of the need for physical machines in Data Centre (DC). Energy consumption by DC is represented between 1.1% and 1.5% of the total world-wide electric power consumption in 2010 [16]. This corresponds to the typical yearly electricity consumption of 120 million households, producing negative greenhouse effects and $CO_2$ footprints. As a result, energy consumption increases more and more rapidly which can threat the environment and cause serious problems. A way to help with this issue is finding energy efficient techniques and algorithms to manage computing resources [3-5, 8, 11, 14].

F e l l e r, R i l l i n g and M o r i n [6] showed a resource allocation based on a homogeneous platforms model to provide the number of physical machines to virtual machine and proved that the energy consumption of the system is reduced if the number of used physical machines is diminished. We will explore the heterogeneous platforms as far as the resources of the physical machines are not the same [12, 13]. In [12, 25] the resource allocation problem is established and FFD algorithm is used to solve it. However, S e t z e r and S t a g e [11] demonstrated that this algorithm tends to wasting capabilities. According to some reviews [3-5, 8, 11, 14], some practices of system resources allocation with the minimal energy consumption are pointed out and are just focused on power utilization on the CPU of physical machines. We believe that this burning is not only upon CPU but also over other appliances such as hard disk, bandwidth, etc.

Therefore, this study aims to solve the problem of resource allocation (physical resources) for energy efficiency by virtual services on the heterogeneous platforms, which involves minimizing the system's energy consumption. The main results are as follows:

- Building the approaches of capability demand, resource allocation, and energy consumption for virtual service from heterogeneous shared hosting Platforms with condition that each virtual service will be a single virtual machine.
- Stating the resource allocation problem for virtual services as an optimal problem: minimizing the energy consumption of the system.
- Developing the ECRA-SA algorithm, which is based on the Simulated Annealing algorithm [13], and using CloudSim tool [3]. Comparison of energy consumption and the execution time between ECRA-SA and FFD algorithm.

The rest of this study is organized as follows: Section 2 presents the related work. Section 3 presents a problem of energy consumption resource allocation. Section 4 proposes the ECRA-SA algorithms to solve the problem. Section 5 follows with experimental results. Section 6 is conclusion and suggests the future work.

## 2. Related work

Resource management for shared hosting platforms has been investigated in many other studies. In particular, U r g a o n k a r, S h e n o y and R o s c o e [20] propose a profiling technique for statistic of resource usage and minimum resource needs; A r o n, D r u s c h e l and Z w a e n e p o e l [19], and C a s a n o v a et al. [23] formulate the resource provisioning problem as a constrained optimization problem, in which machines are considered as a monolithic resource; S t i l l w e l l et al. [22] further consider resource provisioning in a multi-dimension resource; H i e n, F r e d e r i c and M e n a u d [21, 27] addressed the problem of autonomic virtual resource management for hosting service platforms with a two-level architecture, which isolates application specific functions from a generic decision-making layer. However, they focus only on efficiency of resource provisioning. Our study aims to solve the problem of energy efficiency resource allocation.

H e r m e n i e r et al. [24] proposed the Entropy resource manager for homogeneous clusters, which performs dynamic consolidation based on constraint

programming and takes migration overhead into account. However, instead of considering the homogeneous platforms, we will explore the heterogeneous physical machines environment, where resources configuration of the physical machines are not the same.

Jianfang, Junjie and Qingshan [26] used the cloud model and Discrete Particle Swarm Optimization algorithm to quantify the security of the task and virtual machine resources and the security cloud similarity to show the user's security satisfaction.

A number of practices can be applied to achieve energy efficiency. One of them is Dynamic Voltage and Frequency Scaling [2], a mechanism present in modern processors that consists of the combined change of the supply voltage and clock frequency. The main idea is to decrease the voltage and the frequency of the CPU by scaling it down when it is not fully utilized and also to do the opposite actions when it is being entirely utilized. Despite of the improvements in energy efficiency of the hardware, overall energy consumption continues to grow due to increasing requirements for computing resources.

Armbrust et al. [14] studied the problem of request scheduling for multi-tiered web applications in virtualized heterogeneous systems in order to minimize energy consumption while meeting performance requirements. They proposed a heuristic for a multidimensional packing problem as an algorithm for workload consolidation. In previous articles, we have published two algorithms. Which were used to detect deadlock in resources allocation heterogeneous distributed platforms [6, 8]. We provide deadlock detection algorithms and resolve the optimization problems of resources based the recovery of resources allocated. The most of the studies were set to study scheduling policy effectiveness in resources allocation.

Sotomayor [15] proposed a lease-based model and implemented First-Come-First-Serve scheduling algorithm and a greedy based virtual machine mapping algorithm to map leases that include some of virtual machines with/without start time and user duration to a set of homogeneous physical machines. Much of the research conducted homogeneous physical machines.

Along with hardware design, energy efficiency is influenced on how the software manages the resources. Energy efficient resource management techniques were first introduced on mobile devices, where it has direct impact on battery lifetime [2]. These techniques can be adapted for servers and DCs.

Arianyan et al. [1] have presented the strategies of power aware virtual machine placement techniques on a survey. They have discussed the used optimization algorithms to save power. They have classified the energy saving techniques in a Data Centre into static and dynamic methods. They have included in the Static Power Management class and Dynamic Power Management the techniques.

Feller, Rilling and Morin [6] have presented a power aware optimization technique for virtual placement. They applied an ant colony optimization in dynamic workload placement to conserve energy. However, there is a high cost of computation time due to searching for optimum placement.

Therefore, when proposing energy efficient resource allocation, one needs to be aware of the Service Level Agreement (SLA) to avoid performance degradation of

the consumer applications, including increased response times, timeouts or even failures. Therefore, Cloud providers have to establish Quality of Service (QoS) requirements to avoid SLA violations and meeting the QoS requirements while minimizing energy consumption.

## 3. Energy consumption resource allocation problem

### 3.1. System and resource model

A heterogeneous shared platforms 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.

As stated in our previous work [12], for each type of resource under consideration a physical machine may have one or more single element resource (i.e., one or more single real CPU, one or more single real memory, …) and aggregate resources. Thus, the resources of a physical machine are represented by an ordered pair of multi-dimensional resource vectors $(C^e,\ C^a)$. The elementary resource vector $C^e$ gives the capacity of a single element in each dimension while the aggregate resource vector $C^a$ gives the sum of resource capacity counting all elements, in which,
$C^e = \{c_{jk}^e | k = 1,\dots,D; j = 1,\dots,M\}$ and $C^a = \{c_{jk}^a \mid k = 1,\dots,D; j = 1,\dots,M\}$, and $c_{jk}^e$, $c_{jk}^a$ are items which represent the elementary resource and aggregate resource of a physical machine *j* for a resource type *k*, respectively. Inside, *M* be the number of physical machines and *D* be the dimension of resources.

Similarly, 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* [12, 25].

Hence, the rigid needs of a virtual service *i* for a resource type *k* are represented by a first ordered vector pair $(R^e, R^a)$ which determines the resource demands in running the virtual service at the acceptable level, in which, $R^e = \{r_{ik}^e | k = 1,\dots,D; i = 1,\dots,N\}$, and $R^a = \{r_{ik}^a | k = 1,\dots,D; i = 1,\dots,N\}$, and $r_{ik}^e$, $r_{ik}^a$ are items which are to denote the elementary and aggregate rigid needs of a resource type *k* of a virtual service *i*, respectively; *N* is the number of virtual services.

The fluid needs of a virtual service *i* for a resource type *k* are represented by a second ordered vector pair $(F^e, F^a)$, which displays the additional resource demand when running the virtual service at the maximum performance level; in which, $F^e = \{f_{ik}^e | k = 1,\dots,D; i = 1,\dots,N\}$ and $F^a = \{f_{ik}^a | k = 1,\dots,D; i = 1,\dots,N\}$. $f_{jk}^e$, $f_{jk}^a$ are items, which are to denote the elementary and aggregate fluid needs of the resource type *k* of a virtual service *i*, respectively.

The utilization of all resources corresponding to fluid needs is linearly correlated. For example, if the virtual service is only provided by with 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 consumed (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 the virtual service is not provided the fluid resource, whereas with 1 conforming to the virtual service which executes at the maximum resource allocation. Therefore, resource needs of resource type $k$ of virtual service $i$ on physical machine $j$ with additional factor are given by an ordered vector pair $(R^e + Q \times F^e,\ R^a + Q \times F^a)$. In which, $Q = \{q_{ij} | i = 1, \ldots, N; j = 1, \ldots, M\}$ is a vector of additional factor of virtual services when the user requests, and $q_{ij}$ intends the additional factor of virtual service $i$ on physical machine $j$.

### 3.2. 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

$$P_j(u_j) = \left(P_j^{\max} - P_j^{\text{idle}}\right) \times u_j + P_j^{\text{idle}} \quad \forall j. \tag{1}$$

Among them, $P_j^{\max}$ and $P_j^{\text{idle}}$ 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

$$u_j = \sum_{k=1}^{D} \frac{u_{jk}}{c_{jk}^a} = \sum_{k=1}^{D} \frac{\sum_{i=1}^{N} (r_{ik}^a + q_{ij} \times f_{ik}^a) \times x_{ij}}{c_{jk}^a} \quad \forall j. \tag{2}$$

In there, $u_{jk}$ is a resource $k$ of a physical machine $j$ that it is allocated for virtual services, $c_{jk}^a$ is the capacity of aggregate resource $k$ on a physical machine $j$. The physical machines are not used, it will be shut down. Therefore, the energy consumption of the $N$ physical machines in the period $\Delta t$ is set as

$$E(t) = \begin{cases} \Delta t \times \sum_{j=1}^{M} P_j(u_j) & \text{if } u_j \neq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

### 3.3. Objective function and constraints

Assuming that each virtual service consists of a single virtual machine, which has a fixed resource, needs. An Energy Consumption Resource Allocation (ECRA) problem for virtual service is formulated by as follows:

Let $N$ be virtual services, indexed by $i = 1, \ldots, N$; $N > 0$, $M$ be physical machines having the different resource configuration, recorded by $j = 1, \ldots, M$, $M > 0$. Each physical machine provides $D$ types of resource, listed by $k = 1, \ldots, D$; $D > 0$. 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, $P_j^{\max}$ and $P_j^{\text{idle}}$ are the power of physical machine

$j$ in the maximum used utilities state and idle state, respectively. Given these notations, an ECRA problem is described as follows.

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

(5) $$\sum_{j=1}^{M} x_{ij} = 1 \quad \forall i,$$

(6) $$\left(r_{ik}^{e} + q_{ij} \times f_{ik}^{e}\right) \times x_{ij} \ \leq \ c_{jk}^{e} \quad \forall i,j,k,$$

(7) $$\sum_{i=1}^{N} (r_{ik}^{a} + q_{ij} \times f_{ik}^{a}) \times x_{ij} \leq c_{jk}^{a} \quad \forall j,k,$$

(8) $$\left(\sum_{j=1}^{M} \left(\sum_{k=1}^{D} \left((P_j^{\max} - P_j^{\text{idle}}) \times \frac{\sum_{i=1}^{N} (r_{ik}^{a} + Q_{ij} \times f_{ik}^{a})}{c_{jk}^{a}} \times x_{ij} + P_j^{\text{idle}}\right)\right)\right) \times \Delta t \rightarrow \min.$$

Constraint (4) defines the domain of the variables. Constraint (5) determines the state that resources for service $i$ are provided by exactly one physical machine $j$. Constraint (6) states that the elementary resource of the physical machine $j$ does not overcome, and constraint (7) disposes that the aggregate resource of the physical machine $j$ does not overcome. Finally, formulation (8) is the optimized objective to minimum the energy consumption in a time period $\Delta t$.

## 4. Solution based on simulated annealing algorithm

The Simulated Annealing (SA) algorithm [13] is a well studied meta-heuristic. The key feature of simulated annealing is that it provides a mechanism to escape local optima in hopes of finding a global optimum for the optimization problems.

To describe the specific features of a simulated annealing algorithm for discrete optimization problems, several definitions are needed. Let $\Omega$ be the solution space (i.e., the set of all possible solutions). Let $f: \Omega \rightarrow \Re$ be an objective function defined on the solution space. The goal is to find a global minimum, $\omega^*$ (i.e., $\omega^* \in \Omega$ such that $f(\omega^*) \leq f(\omega)$ for all $\omega \in \Omega$). The objective function must be bounded to ensure that $\omega^*$ exists. Define $E(\omega)$ to be the neighbourhood function for $\omega \in \Omega$. Therefore, associated with every solution, $\omega \in \Omega$, are neighbouring solutions, $E(\omega)$, that can be reached in a single iteration of a local search algorithm.

Simulated annealing starts with an initial solution $\omega \in \Omega$. A neighbouring solution $\omega' \in E(\omega)$ is then generated (either randomly or using some pre-specified rule). Simulated annealing is based on the Metropolis acceptance criterion, which models how a thermodynamic system moves from the current solution (state) $\omega \in \Omega$ to a candidate solution $\omega' \in E(\omega)$, in which the energy content is being minimized. The candidate solution, $\omega'$, is accepted as the current solution based on the acceptance probability.

Therefore, 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 loop, the cost function is considered for two solutions: the current solution and the neighbouring solution. If neighbouring solution is better than the current solution, it will be selected. In contrast, the neighbouring 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 $T$. The algorithm will stop after looking for a global optimal value or a temperature as the value $T_{\min}$.

In summary, the pseudo code of ECRA-SA algorithm to solve an ECRA problem is shown as an Algorithm 1.

**Algorithm 1: ECRA-SA**

*Input*:

– Number of virtual services *N*, type of resources *D* and additional factor $q_{ij}$.

– Number of physical machines *M*.

*Output*: List of used physical machines, $S_{\text{best}}$.

**Step 1.** **double** $T \leftarrow T_0$; $T_{\min}$;
**Step 2.** $S_{\text{best}} \leftarrow \text{initSolution}(M, N)$; // *Execute initialization solution by FFD*
**Step 3.** $E_{\text{best}} \leftarrow$ according to Equation (3);
**Step 4.** **while** $(T > T_{\min})$ **do**
**Step 5.** **for** $l := 1$ **to** $L$ **do** // *L is the number of iterative*
**Step 6.** $S_{\text{current}} \leftarrow S_{\text{best}}$;
**Step 7.** $E_{\text{current}} \leftarrow$ according to Equation(3);
**Step 8.** $S_{\text{neighbor}} \leftarrow$ currentNeighborSolution $(M, N)$; // *Execute the current neighbouring solution*
**Step 9.** $E_{\text{neighbor}} \leftarrow$ according to Equation(3);
**Step 10.** **if** $(E_{\text{neighbor}} < E_{\text{current}})$ **then**
**Step 11.** $S_{\text{current}} \leftarrow S_{\text{neighbor}}$;
**Step 12.** $E_{\text{current}} \leftarrow$ according to Equation(3);
**Step 13.** **end if**
**Step 14.** **if**$(\exp(\frac{E_{\text{neighbor}} - E_{\text{current}}}{T}) > \text{random}(0,1))$
**Step 15.** $S_{\text{current}} \leftarrow S_{\text{neighbor}}$;
**Step 16.** $E_{\text{current}} \leftarrow$ according to Equation(3);
**Step 17.** **end if**
**Step 18.** **if** $(E_{\text{current}} < E_{\text{best}})$ **then**
**Step 19.** $S_{\text{best}} \leftarrow S_{\text{current}}$;
**Step 20.** $E_{\text{current}} \leftarrow$ according to Equation(3);
**Step 21.** **end if**
**Step 22.** **end for** $l := 0$ **to** $L$
**Step 23.** $T \leftarrow T \times (1 - \text{CR})$; // *CR is a Cooling Rate*
**Step 24.** **end While**
**Step 25.** **return** $S_{\text{best}}$;

Firstly, we will initialize the initial temperature $T_0$ and the final temperature $T_{\min}$. We obtain the initial solution $S_{\text{best}}$ by FFD [12] and calculate the objective function $E_{\text{best}}$. Next, an initial solution $S_{\text{best}}$ is showed a current solution $S_{\text{current}}$. We generate a feasible neighboring solution, $S_{\text{neighbor}}$, using the random resource allocation method (based on the Algorithm 2) and calculate the objective function, $E_{\text{neighbor}}$, and determine the Metropolis condition (i.e., if $E_{\text{neighbor}} < E_{\text{current}}$ then the new solution, $S_{\text{neighbor}}$, be accepted as the initial solution for next loop. Otherwise, if $(\exp\left(\frac{E_{\text{neighbor}} - E_{\text{current}}}{T}\right) > \text{random}(0,1))$ then new solution $S_{\text{neighbor}}$ can be accepted). If the iteration value *L* is not enough then we continue. Finally,

reduce the temperature $T$, if the new temperature is not greater than the stopping temperature $T_{\min}$ stop and output the optimal solution $S_{\text{best}}$.

**Algorithm 2: CurrentNeighborSolution**()

*Input*:

– Number of virtual services *N*, type of resources *D* and additional factor $q_{ij}$.

– Number of current physical machines $M_{\text{current}}$.

*Output*: List of used physical machines, allocation.

**Step 1. for** $i := 1$ **to** $N$ **do**

**Step 2.** **int** rand $\leftarrow$ **random**($M_{\text{current}}$);

**Step 3.** **for** $j :=$ rand **to** $M_{\text{current}}$ **do**

**Step 4.** **for** $k := 1$ **to** $D$ **do**

**Step 5.** $\text{Sum}_{ik}^{e} \leftarrow r_{ik}^{e} + q_{ij} \times f_{ik}^{e}$ ; // *calculate the total of element resource needs*

**Step 6.** $\text{Sum}_{ik}^{a} \leftarrow r_{ik}^{a} + q_{ij} \times f_{ik}^{a}$ ; // *calculate the total of aggregate resource needs*

**Step 7.** **if**($c_{jk}^{e} \geq \text{Sum}_{ik}^{e}$ **and** $c_{jk}^{a} \geq \text{Sum}_{ik}^{a}$)

**Step 8.** $c_{jk}^{a} \leftarrow c_{jk}^{a} - \text{Sum}_{ik}^{a}$ ;

**Step 9.** **end if**

**Step 10.** **end for** $k := 1$ **to** $D$

**Step 11.** Allocation $[i] \leftarrow j$;

**Step 12.** **break**;

**Step 13.** **end for** $j :=$ rand **to** $M_{\text{current}}$

**Step 14.** **end for** $i := 1$ **to** $N$

**Step 15.** **return** Allocation;

M i t r a et al. [18] have shown that a Simulated Annealing algorithm converges in the limit to a globally optimal solution with probability 1. Therefore, the ECRA-SA algorithm will converge after the finite iteration steps (i.e., $T = T_{\min}$).

Let *N* be the number of virtual services, *M* be the number of physical machines, *D* be the dimension of resources, $T$ be initialized temperature and *L* be the number of loop. The complexity of algorithm is calculated as follows:

- An initializing solution: $O(D \times N \times M)$, a neighbouring solution: $O(D \times N \times M)$, the estimating of cost function: $O(D \times M)$. Therefore, the complexity of algorithms:

$$O(D \times N \times M) + O(T \times L \times (O(D \times N \times M) + O(D \times M))).$$

- If $D, T, L$ are considered as a constant, this algorithm have the computational complexity of $O(N \times M)$.

## 5. Numerical result and evaluation

### 5.1. Simulation

To assess the ECRA-SA algorithm, we compared this algorithm to the FFD [25] by using the CloudSim tool [3]. CloudSim is used to enable modelling and simulation of cloud computing systems and application provision environments. It supports the

modelling of cloud system components such as Data Centre, Hosts, Virtual Machines (VMs) and resource provision policies (Fig. 1).

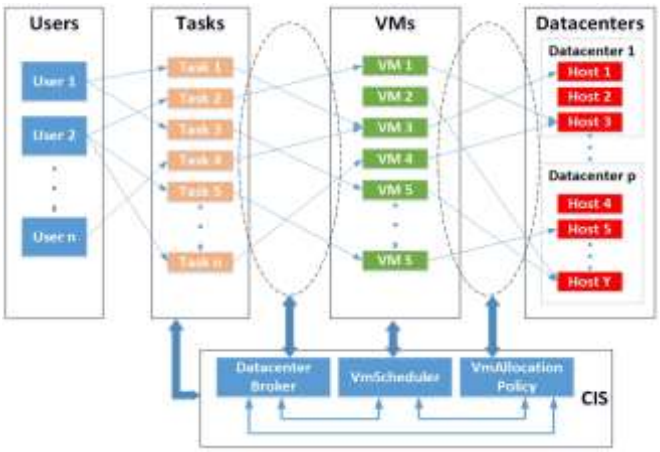


Fig. 1. CloudSim components

As presented in Section 3, each virtual service is one separate virtual machine. Therefore, we inherited the *Vm* layer to extend the resource need features of virtual machine and the *Host* layer to extend the resource features of physical machine. Meanwhile, we inherited the VMAlloctonPolicy layer to perform the resource allocation policy for virtual machine based on the ECRA-SA algorithm. The experimental data are extracted from the real data as indicated in [1, 2]. Table 1 presents the resource characteristics and the power of physical machines. The resource characteristics of virtual machines, which are similar to the virtual machines of Amazon EC2 cloud, are adjusted to suit the problem, shown in Table 2.

Table 1. Type of physical machine

| Type of physical machine | CPU (MHz) | RAM (GB) | BW (GB per 1 s) | Disk (GB) | $P^{idle}$ (kW) | $P^{max}$ (kW) |
|---|---|---|---|---|---|---|
| HP proliant G4 | 2 core × 1860 | 4 | 1 | 20 | 86 | 117 |
| HP proliant G5 | 2 core × 2660 | 4 | 1 | 40 | 93.7 | 135 |
| IBM Server × 3250 | 4 core × 2933 | 8 | 1 | 600 | 46.1 | 113 |
| IBM Server × 3550 | 6 core × 3067 | 16 | 1 | 800 | 58.4 | 222 |

Table 2. Type of virtual machine

| Type of virtual machine | CPU(MHz) | RAM(GB) | BW (GB per 1 s) | DISK(GB) |
|---|---|---|---|---|
| Virtual Machine 1 | 2500 | 0.85 | 0.45 | 5 |
| Virtual Machine 2 | 2000 | 3.75 | 0.35 | 10 |
| Virtual Machine 3 | 1000 | 1.7 | 0.25 | 15 |
| Virtual Machine 4 | 500 | 0.613 | 0.15 | 20 |

## 5.2. The experimental results

The parameters of the ECRA-SA algorithms are used: $T_0$=1000; CR=5; $T_{min}$=0; $L$=100. The number of virtual machines (represents for the number of virtual services) in the experimental is 100; 200; 300; 400; 500 respectively and the energy

consumption in a time period of $\Delta t$ = 24 hours. Measure unit of energy consumption is kW.h, and the runtime of algorithm is calculated by second (s). The performed time is measured by a PC having an Intel(R) Core(TM) i5–3235M 2.60 GHz, RAM 4Gb microprocessor. Moreover, we use the next formula to compare the gain energy consumption between two algorithms:

$$\text{Gain Energy (\%)} = \frac{E^{\text{FFD}} - E^{\text{ECRA-SA}}}{E^{\text{ECRA-SA}}} \times 100, \quad (9)$$

where $E^{\text{FFD}}$ is the energy consumption when using the FFD algorithm and $E^{\text{ECRA-SA}}$ is the energy consumption when using the ECRA-SA algorithm, respectively.

Table 3. Used physical machines and consumed energy

| No of VM | Algorithm | Times (s) | Energy (kW.h) | Gain Energy (%) |
|---|---|---|---|---|
| 100 | FFD | 0.031 | 201.284 | |
| | ECRA-SA | 0.038 | 193.000 | 4.292 |
| 200 | FFD | 0.078 | 396.706 | |
| | ECRA-SA | 0.088 | 392.490 | 1.074 |
| 300 | FFD | 0.116 | 597.989 | |
| | ECRA-SA | 0.125 | 577.754 | 2.297 |
| 400 | FFD | 0.144 | 793.411 | |
| | ECRA-SA | 0.160 | 772.185 | 2.749 |
| 500 | FFD | 0.200 | 994.694 | |
| | ECRA-SA | 0.218 | 972.439 | 2.289 |

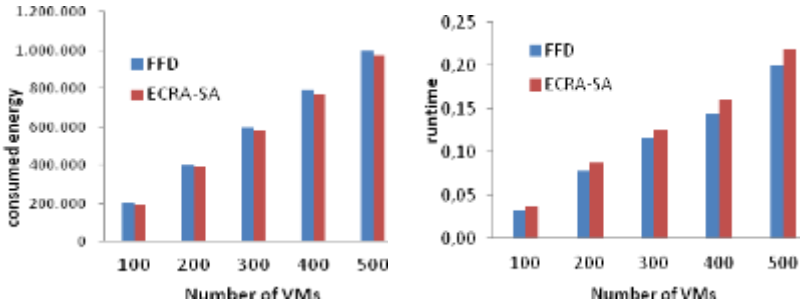


Fig. 2. The graph of runtime and consumed energy

Based on the results in Table 3 and Fig. 2, we showed that consumed energy of ECRA-SA algorithm is better than FFD. Because the FFD algorithm tends to use a large number of physical machines, the ECRA-SA algorithm uses a more efficient global search solution. However, the execution time of the ECRA-SA algorithms are larger than FFD algorithm. A reason of this is affected by parameter of the number of iterations in ECRA-SA algorithms, which provides a mechanism to escape local optima in hopes of finding a global optimum for the optimization problems. However, the execution time of the algorithm is short and it can be applied in practice.

The ECRA-SA algorithm could give optimal or non-optimal results depending on the input value of the algorithm, including the number of virtual services *N*, the number of resource types *D*, rigid needs of virtual service and the resource capacity of physical machines.

If the resource capacity of the physical machine is strong enough and the resource needs of the virtual service are low, the ECRA-SA algorithm gives a good effective. The best case of the algorithm is that when the first loop an acceptable solution (the value of objective function is a best value), then the algorithm stops and offers the acceptable solution.

The worst case is that finished all loops that do not find any solutions and the execution time of the algorithm is very large because of the algorithm complexity. This case occurs when the resource needs of virtual service are too large, and the system does not have enough resources to provide.

## 6. Conclusion

The article presents the Multi-Objective Resource Allocation problem for virtual services with the optimal constraints: minimize the energy consumption of the system. We propose the ECRA-SA algorithm based on the Simulated Annealing method. We also set up, assessed and compared it with the FFD algorithm through the parameters consumed energy and execution time. The algorithms were executed on the real data by CloudSim tool. The experimental outstanding shows that the energy consumption proposed by the ECRA-SA algorithm is better than the performance of FFD algorithm. For the future work, the proposed approach will be extended for multi-objective dynamic resource allocation.

## References


1. Arianyan, E., et al. Novel Energy and SLA Efficient Resource Management Heuristics for Consolidation of Virtual Machines in Cloud Data Centers. – Computers Electrical Engineering, Vol. **47**, 2015, pp. 222-240.
2. Beloglazov, A., 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, Vol. **24**, 2012, No 13, pp. 1397-1420.
3. Calheiros, R. N., et al. Cloudsim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. – Softw. Pract. Exper. Vol. **41**, 2011, No 1, pp. 23-50.
4. Cao, Z., S. Dong. Dynamic VM Consolidation for Energy-Aware and SLA Violation Reduction in Cloud Computing. – In: Proc. of 2012 IEEE International Conference on Parallel and Distributed Computing, 2012. PDCAT, IEEE, 2012, pp. 363-369.
5. Farahnakian, F., et al. Energy-Aware Dynamic VM Consolidation in Cloud Data Centers Using Ant Colony System. – In: Proc. of 2014 IEEE International Conference on Cloud Computing, 2014. CLOUD, IEEE, 2014, pp. 104-111.
6. Feller, E., L. Rilling, C. Morin. Energy-Aware Ant Colony Based Workload Placement in Clouds. – In: Proc. of 2011 IEEE/ACM International Conference on Grid Computing, 2011. GRID, IEEE, 2011, pp. 26-33.
7. Nguyen, H. H. C., et al. A New Technical Solution for Resource Allocation in Heterogeneous Distributed Platforms. – In Advances in Digital Technologies, IOS Press, The Netherlands, University of Macau, Macau, 2015, pp. 184-194.
8. Nguyen, H. H. C., et al. Deadlock Detection for Resource Allocation in Heterogeneous Distributed Platforms. – In: Recent Advances in Information and Communication Technology 2015, Springer, 2015, pp. 285-295.

9. Luo, L., et al. A Resource Scheduling Algorithm of Cloud Computing Based on Energy Efficient Optimization Methods. – In: Proc. of International Green Computing Conference (IGCC'12), 2012, pp. 1-6.
10. Quan, D. M., et al. Energy Efficient Resource Allocation Strategy for Cloud Data Centres. – Computer and Information Sciences II: 26th International Symposium on Computer and Information Sciences, Springer, 2012, pp. 133-141.
11. Setzer, T., A. Stage. Decision Support for Virtual Machine Reassignments in Enterprise Data Centers. – In: Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP, 2010, pp. 88-94.
12. Pham, N. M. N., et al. Resource Allocation for Virtual Service Based on Heterogeneous Shared Hosting Platforms. – In: Proc. of 8th Asian Conference, 2016. ACIIDS, Springer, 2016, pp. 51-60.
13. Kirkpatrick, S. Optimization by Simulated Annealing: Quantitative Studies. – Journal of Statistical Physics, Vol. **34**, 1984, No 5, pp. 975-986.
14. Armbrust, M., et al. Above the Clouds: A Berkeley View of Cloud Computing. – Technical Report No UCB EECS-2009-28, University of California at Berkley, USA, 2009.
15. Sotomayor, B. Provisioning Computational Resources Using Virtual Machines and Leases. PhD Thesis Submited to University of Chicago, USA, 2010.
16. Koomey, J. Growth in Data Center Electricity Use 2005 to 2010. – A Report by Analytical Press, Completed at the Request of New York Times, Vol. **9**, 2011.
17. Ranjana, R., J. Raja. A Survey on Power Aware Virtual Machine Placement Strategies in a Cloud Data Center. – In: Proc. of 2013 International Conference on Green Computing, 2013. ICGCE, IEEE, 2013, pp. 747-752.
18. Mitra, D., F. Romeo et al. Convergence and Finite-Time Behavior of Simulated Annealing. – Advances in Applied Probability, Vol. **18**, 1986, No 3, pp. 747-771.
19. Aron, M., P. Druschel, W. Zwaenepoel. Cluster Reserves: A Mechanism for Resource Management in Cluster-Based Network Servers. – In: Proc. of 2000 ACM Sigmetrics International Conference on Measurement and Modeling of Computer Systems, New York, USA, 2000, pp. 90-101.
20. Urgaonkar, B., P. Shenoy, T. Roscoe. Resource Overbooking and Application Profiling in Shared Hosting Platformsss. – SIGOPS Operating Systems Review, Vol. **36**, 2002, No SI, pp. 239-254.
21. Hien, N. V., D. T. Frederic, J. M. Menaud. SLA-Aware Virtual Resource Management for Cloud Infrastructures. – In: Proc. of 9th IEEE International Conference on Computer and Information Technology (CIT'09), Xiamen, China, 2009, pp. 1-8.
22. Stillwell, M., D. Schanzenbach, F. Vivien, H. Casanova. Resource Provisioning Algorithms for Virtualized Service Hosting Platformsss. – Journal Parallel Distrib. Comput., Vol. **70**, 2010, pp. 962-974.
23. Casanova, H., D. Schanzenbach, M. Stillwell, F. Vivien. Resource Provisioning Using Virtual Clusters. – Research Report No 2008-33, October 2008.
24. Hermenier, F., et al. Entropy: A Consolidation Manager for Clusters. – In: Proc. of 5th International Conference on Virtual Execution Environments (VEE'2009), March 2009, Washington, DC, United States. ACM, ACM SIGPLAN/SIGOPS, 2009, pp. 41-50.
25. Stillwell, M., F. Vivien, H. Casanova. Virtual Machine Resource Allocation for Service Hosting on Heterogeneous Distributed Platforms. – In: Proc. of 2012 IEEE 26th International Parallel and Distributed Processing Symposium, IPDPS'12, IEEE Computer Society, Washington, DC, USA, 2012, pp. 786-797.
26. Jianfang, C., C. Junjie, Z. Qingshan. An Optimized Scheduling Algorithm on a Cloud Workflow Using a Discrete Particle Swarm. – Cybernetics and Information Technologies, Vol. **14**, 2014, No 1, pp. 25-39.
27. Hien, N. V., D. T. Frederic, J. M. Menaud. Autonomic Virtual Resource Management for Service Hosting Platforms. – In: Proc. of 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing. IEEE Computer Society, 2009, pp. 1-8.