Computing and Informatics, Vol. 37, 2018, 76–108, doi: 10.4149/cai_2018_1_76

ENERGY AWARE RESOURCE ALLOCATION FOR CLOUDS USING TWO LEVEL ANT COLONY OPTIMIZATION

Ashok Kumar, Rajesh Kumar, Anju Sharma

Department of Computer Science & Engineering
Thapar University
Patiala-147004, India
e-mail: `ashok.khunger@gmail.com, {rakumar, anju.sharma}@thapar.edu`

Abstract. In cloud environment resources are dynamically allocated, adjusted, and deallocated. When to allocate and how many resources to allocate is a challenging task. Resources allocated optimally and at the right time not only improve the utilization of resources but also increase energy efficiency, provider's profit and customers' satisfaction. This paper presents ant colony optimization (ACO) based energy aware solution for resource allocation problem. The proposed energy aware resource allocation (EARA) methodology strives to optimize allocation of resources in order to improve energy efficiency of the cloud infrastructure while satisfying quality of service (QoS) requirements of the end users. Resources are allocated to jobs according to their QoS requirements. For energy efficient and QoS aware allocation of resources, EARA uses ACO at two levels. First level ACO allocates Virtual Machines (VMs) resources to jobs whereas second level ACO allocates Physical Machines (PMs) resources to VMs. Server consolidation and dynamic performance scaling of PMs are employed to conserve energy. The proposed methodology is implemented in CloudSim and the results are compared with existing popular resource allocation methods. Simulation results demonstrate that EARA achieves desired QoS and superior energy gains through better utilization of resources. EARA outperforms major existing resource allocation methods and achieves up to 10.56 % saving in energy consumption.

Keywords: Energy efficiency, resource allocation in cloud, dynamic voltage frequency scaling, ant colony optimization, quality of service

1 INTRODUCTION

Cloud computing is a paradigm that has huge potential in enterprise and business. It has a large pool of configurable resources which can be acquired and used on demand [1, 20]. The acquired resources can be accessed over the network. In cloud, everything is provided as a service. Cloud has three service models, namely: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). In IaaS, fundamental computing resources like processing, storage, networks, etc. are provisioned to the consumers for deployment and execution of arbitrary software [20]. The resources are provisioned and allocated according to consumers' demands. Furthermore, resource allocation mechanism is to guarantee that requirements of all applications are suitably met. Due to all these reasons, resource allocation in cloud computing is one of the important challenges. Apart from resource allocation for performance, i.e., allocating sufficient resources to user applications in order to satisfy QoS parameters, another challenge posed to researchers and industry is to minimize the energy consumption and carbon footprints. According to Koomey [18]: *"Total data center power consumption from servers, storage, communications, cooling, and power distribution equipment accounts for 1.7–2.2 % of total electricity used in U.S. in 2010"*. With their enormous appetite for energy, today's data centers emit as much carbon dioxide as whole of Argentina. If left on their current path, data center carbon dioxide output will quadruple by the year 2020 [15]. While the cloud energy appetite is growing quickly, industrial organizations and researchers are finding ways to reduce the energy consumption. Several methods to reduce the energy consumption of a data center exists. Data centers infrastructure is generally over-provisioned to sustain availability of resources during peak hours. But due to dynamic nature of load average resource utilization is approximately 15–20% [26, 15]. Energy efficiency can be improved by better management of resources. One such area for reduction in energy consumption is efficient resource allocation. A large extent of energy can also be saved by server consolidation and turning off idle servers. Sometimes consolidation is not economically feasible due to constraints such as communication cost of migration, QoS violations due to interruption in service while consolidating, or unavailability of PM with sufficient free resources where the VM can be migrated. In such cases energy consumed by PMs can be saved by adjusting its operating voltage/frequency.

In this paper, we proposed EARA methodology that uses ACO for resource allocation. Resources are allocated to the jobs with the goal to minimize total cost of execution, total execution time and total energy consumption while satisfying QoS requirements of the end users. Each QoS parameter of the job is associated with some weight value that indicates its priority over the others. ACO is applied at two levels for efficient allocation of resources. The first level ACO allocates VM resources to jobs whereas the second level ACO allocates PM resources to VMs. Server consolidation and dynamic performance scaling is employed to conserve energy. Dynamic performance scaling is used when server consolidation is economically unfeasible because of high communication cost, QoS violations due to interruption in service or unavailability of destination machine with sufficient free resources. The proposed methodology is implemented in CloudSim and its effectiveness is evaluated with jobs having different resource demands and QoS requirements.

The rest of the paper is organized as follows: Related work is presented in Section 2. Section 3 discusses the energy aware resource allocation methodology and its mathematical representation. Section 4 explains the technique used for EARA that is ant colony optimization. Comparative performance analysis of EARA with the first fit decreasing (FFD), and multi-objective grouping genetic algorithm (MGGA) is presented in Section 5. Conclusion and the scope of future work is detailed in Section 6.

2 RELATED WORK

Beloglazov et al. [3] proposed power efficient and QoS aware resource allocation heuristics. An algorithm for minimization of number of VM migrations is also proposed. Upper and lower threshold utilization levels are set to detect overloaded and underloaded machines. When the resource utilization of a particular server falls below the lower threshold value, all the VMs running on the machine are shifted to some other machine. If utilization of a machine is above upper threshold, one or more VMs are shifted to other machines to keep the utilization between the threshold values. They proposed algorithms for single core machines. In real cloud environment heterogeneous multicore systems are used. Gao et al. [11] proposed the linear programming based multi-objective ant colony based system for virtual machine placement to minimize resource wastage and power consumption. Initial pheromone value is assigned to VM-host movement. The pheromone value indicates probability of a host to be selected for allocation of VM under consideration. The authors used only CPU processing speed and memory requirements of VM while allocating resources to the VMs. Kinger et al. [17] proposed event driven prediction based proactive temperature aware VM scheduling to keep temperature of a server below the specified upper threshold temperature. Temperature predictor constantly monitors temperature of the physical machine. The authors used "unified list" to store current as well as threshold temperature of each node. The unified list is updated after a fixed interval of time, which would cause network congestion, performance degradation and limited scalability. Quarati et al. [23] proposed two level brokering algorithm for hybrid cloud with the objective to maximize broker's revenue and user satisfaction. First level scheduler schedules the requested services on private or public cloud based on reserved quota of private resources. The authors proposed three first level scheduling techniques namely feasible, static reservation, and maximum occupation. Second level of scheduling uses less consuming resource and dynamic less consuming resource techniques to allocate resources to the services. The requested services are run on physical machine having maximum availability of free resources. The proposed technique causes uneven distribution of workload among the servers and overloading of high performance machines. Overloading results in creation of hot spots and increase in rate of failure. Lee et al. [19] proposed performance analysis based resource allocation strategy for green cloud. Every PM in a data center is assigned a performance value based on CPU processing speed, number of cores, and memory capacity relative to machine having maximum number of cores, CPU processing speed, and memory capacity. A PM is allocated to a VM if its performance value fits best the VM requirements. The proposed method results in hot spots and in overloading of high performance machines. The improper distribution of load among servers would cause wastage of energy. Raycroft et al. [24] analyzed the effect of global VM allocation policy on energy consumption. Simulation is performed for the same type of applications but real cloud hosts diverse type of applications. Communication cost between VMs and QoS is not taken into account. Moreover, the authors proposed movement of VMs between regions which is impractical in case of large sized VM. Feller et al. [10] proposed multi-dimensional ant colony optimization based workload consolidation algorithm. The algorithm uses resource utilization history to predict future resource demands and dynamically overbooks the resources. The authors have tested the algorithm on PM having the same capacity, i.e. in homogeneous environment. Real cloud environment is heterogeneous in nature, having machines with different resource capacity. Gao et al. [11] proposed multi-objective ant colony system algorithm for virtual machine placement that minimizes total resource wastage and power consumption. The algorithm attempts to utilize server to its full capacity which would result in creation of hot spots and increase in number of service-level agreement (SLA) violations. Moreover, using server near full capacity causes more heat dissipation which results in decrease in server reliability. Nathani et al. [21] proposed modified immediate and advance reservation algorithms for deadline sensitive leases. The proposed algorithms try to schedule new lease as a deadline sensitive lease in a single or multiple time slots. If a new lease cannot be scheduled in a single or multiple time slots, the algorithm reschedules the already scheduled deadline sensitive leases to make a room for new lease. In case, rescheduling fails to generate deadline constrained schedule, then backfilling is applied to accommodate new lease. The drawback of the proposed algorithm is its high lease preemption rate which, in turn, increases the allocation overhead. Ant colony optimization technique for assigning real-time tasks to heterogeneous processors is proposed by Chen et al. [7]. Local search technique is applied to improve energy efficiency of the feasible assignment solution generated by the proposed assignment algorithm. The authors have claimed that their algorithm saves 15.8 % energy over prototyped version of ant colony optimization. Huang et al. [14] proposed adaptive sub-optimal resource management scheme. In the proposed scheme, global resource allocation module uses remaining resource table and resource utilization rate table to estimate number of VMs required to provide desired level of service. Genetic algorithm (GA) is proposed for reallocation of resources to achieve better performance. The proposed technique suffers from a single point failure. Moreover, centralized global resource allocation module, remaining resource table, and resource utilization rate table will cause performance degradation when the number of requests is large. In [6, 16], the authors proposed methods to calculate energy consumption of a PM. Castañé et al. [6] modeled four basic subsystems: computing, memory, storage, and network of a PM to calculate the amount of energy consumed by it. Kim et al. [16] proposed a methodology for estimating energy consumption of a VM based on its in-processor events without using a dedicated energy measuring instrument. Performance counters available in modern processor are used to keep the track of specific type floating point instructions issued by VM. Based on the instruction type and its count, the energy consumption of VM is estimated. The number of performance counters available in a processor is limited, so limited number of instructions can be tracked, and that results in erroneous estimation of energy consumption. The authors also proposed the energy credit scheduler. The proposed scheduler assigns resources to the VM based on its energy credit. The resources allocated to VM are preempted when its energy credit vanishes. Garg et al. [12] proposed green cloud computing framework for reducing carbon footprint without sacrificing QoS. The authors used Green Offer Directory and Carbon Emission Directory to offer green services to the users. The Carbon Emission Directory maintains data related to the energy efficiency of cloud services. Based on the information in these two directories, the cost and carbon footprint of leasing a particular cloud service are calculated. The providers are supposed to publish carbon footprint and energy efficiency of their services in public directories. There is no check on the data that is published by the providers. The service provider can publish manipulated data in order to earn more and for building its reputation in the market. Xu and Fortes [28] proposed multi-objective VM allocation algorithm. The authors have taken CPU, and memory parameters for VMs and have claimed reduction in power consumption, thermal dissipation costs, and resource wastage. Disk utilization and inter VM communication cost is not taken into consideration. Wu et al. [27] proposed energy efficient priority job scheduling for cloud computing. The requirements of a job are given in terms of maximum and minimum CPU frequencies. Every server is assigned some weight based on its performance/Watt. Servers are selected for jobs according to assigned weight and SLA level required by the the users. A job is assigned to VM running on a selected server that meets its requirements. Frequency of the server is then tuned to reduce energy consumption. However, the authors have not considered memory, input/output and other requirements of the job. In [25, 4, 22, 13], the authors proposed energy-conscious consolidation heuristics in order to conserve energy and maximize resource utilization without affecting the performance of the system. Takeda and Takemura [25] proposed ranking of physical servers for consolidation and VM placement. Servers with higher priorities are considered more reliable than the servers with lower priority value. Higher priorities are assigned to newly installed servers. The main drawback of the server ranking strategy is that the priorities are to be assigned to the server by the operator manually.

3 ENERGY AWARE RESOURCE ALLOCATION METHODOLOGY

The proposed EARA methodology allocates resources to jobs using ant colony optimization. It utilizes the resources efficiently to save energy besides fulfilling the operational demands of the jobs. Each job has some resource and QoS requirements. Each QoS parameter of a job is associated with a weight value. EARA allocates resources to jobs in accordance with their resources demands and weight values of QoS parameters. The key features of proposed EARA are:

- It deliberately assigns resources to jobs in order to improve the utilization of resources, thereby increases the energy efficiency of the cloud infrastructure.
- Idle PMs are switched to sleep mode to conserve energy.
- Monitoring of utilization of resources viz. processor, memory, network bandwidth of a PM for energy efficient resource allocation and management.
- Dynamic performance scaling of servers to conserve energy.
- Server consolidation to minimize number of active servers.

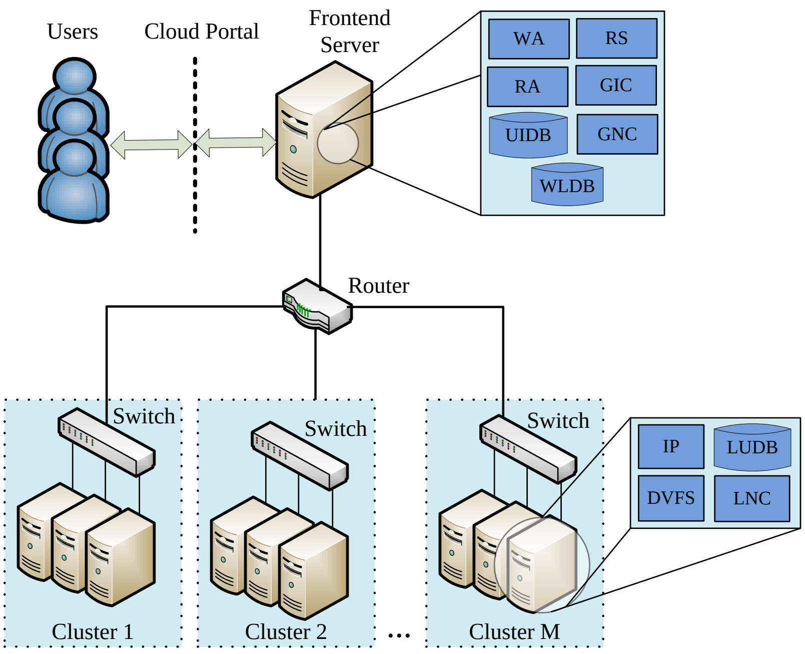


Figure 1. Energy aware resource allocation

The various components of EARA as shown in Figure 1 are:

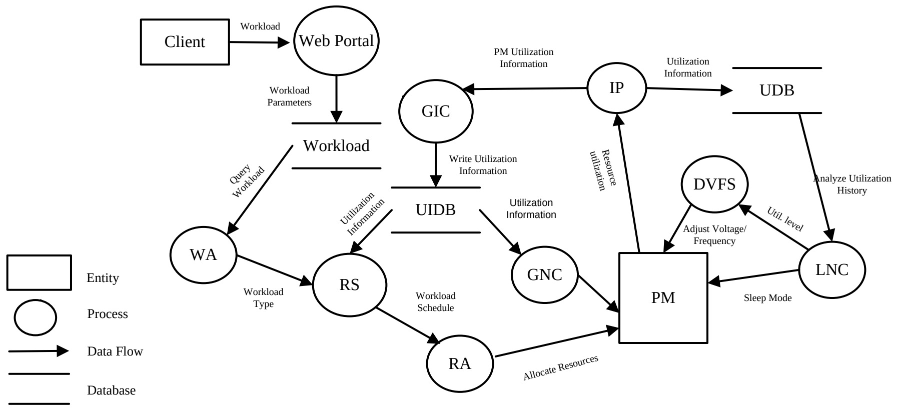


Figure 2. Data flow representation for energy aware resource allocation methodology

Cloud Portal: It provides an interface to the cloud users to input their jobs and desired QoS.

Workload Analyzer (WA): It analyses QoS requirements of the jobs and classifies them into different classes using k-means cluster algorithm.

Resource Scheduler (RS): It generates schedule of jobs to be executed.

Resource Allocation (RA): It applies ant colony optimization to allocate jobs to VMs and VMs to PMs. Resources are allocated in accordance with resource demands and weight values of QoS parameters associated with a job.

Global Information Collector (GIC): It receives the resource utilization data from information probes (IP) of every PM and stores it in utilization information database (UIDB).

Utilization Information Database (UIDB): Resource utilization data of every PM is kept in UIDB for future resource allocation and VM migration decisions.

Global Node Controller (GNC): It initiates live migration of VMs running on a PM when resource utilization of PM violates lower green threshold (LGT) or upper green threshold (UGT) limit.

Workload Database (WLDB): It stores the information associated with each job.

Information Probes (IP): It monitors the utilization of resources viz. processor, memory, network bandwidth of a PM and records observed values in local utilization database (LUDB).

Local Utilization Database (LUDB): It keeps record of utilization of resources of PMs.

Dynamic Voltage Frequency Scaling (DVFS): It adjusts the voltage and frequency of the PM in order to save power and to reduce heat dissipation. The voltage/frequency of PM is adjusted in accordance to resource demands of the VMs/jobs running over it.

Local Node Controller (LNC): It switches the PM to sleep mode if it is found idle for specific period of time.

The overview of various components of EARA and information flow among them is depicted through data flow diagram using Yourdon/DeMarco notation [35] (Figure 2). The directional lines (arrows) show information exchange between components. They do not give any information about timing and sequence of execution of processes. The client inputs jobs from cloud portal. The frontend server stores the job's requirements in workload database for analyzing, energy efficient scheduling, and allocation. Each job has a different resource and QoS requirements. For example, batch job may require storage and computing resources (i.e. memory, CPU), whereas for online job, network bandwidth may be more critical. WA analyses the jobs and classifies them into different categories based on weight values of their QoS parameters. Jobs are then mapped to VMs. Resources are allocated to VMs and then scheduled on PMs. Once the VM is deployed on the PM, its resource utilization is monitored by IP after a customizable fixed interval and observed values are stored in LUDB. When utilization of PM remains below the lower green threshold (LGT) for two consecutive monitoring intervals, either of the two methods is applied to save energy consumed by a PM. First, offloading PM by migrating the VM running on it to some other PM and then switching it to sleep mode. Sometimes, it is economically unfeasible to migrate VM running on PM to some other PM, either due to high migration cost, or deadline violation due to migration, or unavailability of PM that can host the VM due to insufficient available free resources. When VM migration is not economically feasible, the second method, that is dynamic performance scaling of PM, is applied. In dynamic performance scaling, voltage/frequency of the PM is intentionally varied to change its performance. DVFS helps to cut power cost but at the price of a slower job execution. It is applied when the user deadlines can be achieved at slower execution speed.

For implementation purposes and subsequent evaluation of EARA, mathematical equations for energy aware resource allocation, DVFS, and fitness function are formulated as discussed in the forthcoming subsections.

3.1 Mathematical Modeling of Energy Aware Resource Allocation

EARA contemplates the energy efficient usage of resources. EARA is considered from both the provider's and clients' point of view. It minimizes:

1. total energy consumption for the benefit of service provider, and
2. total execution cost and total time of execution for the end users' satisfaction.

The following assumptions are taken into considerations while formulating mathematical representation of EARA.

1. Physical nodes can be unilaterally switched on/off, or put to sleep mode as and when required.
2. Every PM supports advanced configuration and power interface (ACPI). The operating system can adjust voltage and frequency to any level supported by the hardware.
3. PMs and VMs are characterized by processing speed, memory, and network bandwidth.
4. Transition from one voltage/frequency level to other is instant.
5. Energy consumed by PM during sleep mode is negligible.
6. All the cores of a PM can be operated on the same voltage/frequency level at a time.

Total power consumption of a PM [2] at any instant is given by Equation (1).

$$\begin{aligned} P_{total} &= P_{dynamic} + P_{static} \\ &= \underbrace{ACV^2 f}_{\text{DPC}} + \underbrace{V * I_{leak}}_{\text{SPC}}, \end{aligned} \tag{1}$$

where A is switching activity, C is capacitance, V is voltage, I_{leak} is the leakage current, and f is the clock frequency applied to the cores of PM. P_{total}, P_{static}, and $P_{dynamic}$ are total power consumption, static power consumption (SPC), and dynamic power consumption (DPC) of a PM, respectively. SPC is due to leakage current that is present in any active circuit, and it is independent of clock frequency and usage scenario. It can be reduced by switching PM to sleep mode [2]. Whereas, DPC is due to circuit activity and it depends on usage scenario or resource utilization. EARA reduces SPC by switching PM to sleep mode as and when required, whereas DPC is optimized by efficient utilization of resources as explained below.

Suppose s is processing speed, and p is DPC of a PM. Then, $s \propto f$, and $f \propto V$, which implies $p \propto f^3$ and $p \propto s^3$ [34]. Suppose operational requirements of N_t tasks (jobs) can be fulfilled by N_v number of VMs, and each VM has resources to fulfill needs of n_g tasks. R_g is execution requirement and E_{ijg} total energy consumption of the tasks deployed on VM g that is instantiated on PM j of cluster i. If r_{ij}^q is the execution requirement of task q on this PM, then the execution time t_{ij}^q of the task q on this PM with power p_{ij}^q and processing speed s_{ij}^q can be calculated from Equation (2).

$$t_{ij}^q = \frac{r_{ij}^q}{s_{ij}^q} = \frac{r_{ij}^q}{\left(p_{ij}^q\right)^{\frac{1}{3}}}, \tag{2}$$

and the energy consumed by this PM to execute task is given by e_{ij}^q as shown in Equation (3).

$$e_{ij}^q = p_{ij}^q . t_{ij}^q = r_{ij}^q \left(p_{ij}^q\right)^{\frac{2}{3}} = r_{ij}^q \left(s_{ij}^q\right)^2 . \tag{3}$$

With time constraint T_g, the objective is:

1. to minimize energy consumed (E_{ijg}) by VM g, and
2. the execution time of all tasks T_{ijg}= $t^1_{ij}+t^2_{ij}+\ldots+t^{n_g}_{ij}$ should not exceed deadline time T_g.

Energy consumption (E_{ijg}) and execution time (T_{ijg}) are calculated as shown in Equations (4) and (5).

$$E_{ijg}\left(p^1_{ij}, p^2_{ij}, \ldots, p^n_{ij}\right) = r^1_{ij} {p^1_{ij}}^{\frac{2}{3}} + r^2_{ij} {p^2_{ij}}^{\frac{2}{3}} + \ldots + r^{n_g}_{ij} {p^{n_g}_{ij}}^{\frac{2}{3}}, \tag{4}$$

and

$$T_{ijg}\left(p^1_{ij}, p^2_{ij}, \ldots, p^n_{ij}\right) = \frac{r^1_{ij}}{{p^1_{ij}}^{\frac{1}{3}}} + \frac{r^2_{ij}}{{p^2_{ij}}^{\frac{1}{3}}} + \ldots + \frac{r^{n_g}_{ij}}{{p^{n_g}_{ij}}^{\frac{1}{3}}} \leq T. \tag{5}$$

Both the energy consumption and execution time are functions of $p^1_{ij}, p^2_{ij}, \ldots, p^{n_g}_{ij}$. So, solving Equations (4) and (5) using Lagrange multiplier system [37], we get execution time as represented in Equation (6), and energy consumption, as shown in Equation (7).

$$T_{ijg} = \frac{R_g^{\frac{3}{2}}}{\sqrt{E_{ijg}}}, \tag{6}$$

$$E_{ijg} = P_{ijg} T_{ijg} = \left(\frac{R_g}{T_{ijg}}\right)^3 T_{ijg} = \frac{R_g^3}{(T_{ijg})^2} \tag{7}$$

where $R_g = r^1_{ij} + r^2_{ij} + \ldots + r^{n_g}_{ij}$ is the total execution requirement of all the tasks deployed on VM g.

Total energy consumption (TEC) by all the VMs can be calculated as shown in Equation (8).

$$TEC = \sum_{g=1}^{N_v} E_{ijg}. \tag{8}$$

Total execution time (TET) of N_t tasks running on N_v VMs is calculated as shown in Equation (9).

$$TET = T_{ij1} + T_{ij2} + \ldots + T_{ijN_v}. \tag{9}$$

In order to minimize TET and TEC, every VM has to be carefully mapped to suitable PM of a cluster.

A set $CL = \{cl_i \mid 1 \leq i \leq N_c\}$ of clusters is considered. Each cluster i has a pool of physical machines $PM_i = \{h_{ij} \mid 1 \leq j \leq H_i\}$. Here, PM j of cluster i has computing power represented by $h_{ij} = \{hs_{ij}, hm_{ij}, hn_{ij}\}$, where hs_{ij}, hm_{ij}, and hn_{ij} are CPU processing speed, memory, and network bandwidth, respectively. VM computing requirements are represented by $v_g = \{vs_g, vm_g, vn_g\}$, where vs_g, vm_g, and vn_g are processing speed, memory, and network bandwidth of VM g, respectively. The symbolic notations used in mathematical formulation of EARA are depicted in Table A1 of Appendix A.

Suppose C_{ijg} is total cost (Processing cost + Memory cost + Bandwidth cost) of running VM g on PM j of cluster i, x_{ijg} equal to 1 if VM g is assigned to PM j of cluster i and 0 otherwise.

Execution cost (EC_{ij}) of all the VMs running on PM j of cluster i can be calculated from Equation (10):

$$EC_{ij} = C_{ijg} * x_{ijg}, \forall g | x_{ijg=1}, \tag{10}$$

and total cost of execution (COE) of all the VMs can be evaluated using Equation (11):

$$COE = \sum_{i=1}^{N_c} \sum_{j=1}^{H_i} EC_{ij}. \tag{11}$$

A VM should be assigned to PM of a cluster in such a way that all the three conditions, represented by Equations (12), (13), and (14) are satisfied.

$$\sum_{g=1}^{N_v} vs_{ijg} * x_{ijg} \leq hs_{ij} * CPU_{UGT}, \qquad \forall i, j | x_{ijg} = 1, \tag{12}$$

$$\sum_{g=1}^{N_v} vm_{ijg} * x_{ijg} \leq hm_{ij} * MEM_{UGT}, \qquad \forall i, j | x_{ijg} = 1, \tag{13}$$

$$\sum_{g=1}^{N_v} vn_{ijg} * x_{ijg} \leq hn_{ij} * BW_{UGT}, \qquad \forall i, j | x_{ijg} = 1. \tag{14}$$

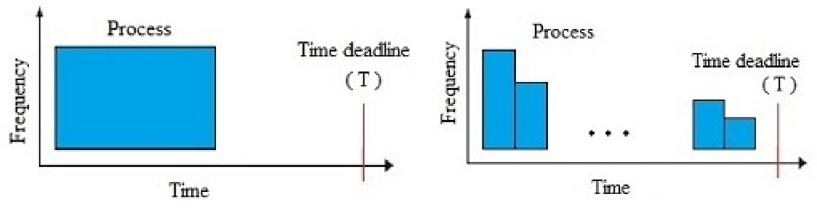
These three conditions put a cap on the maximum utilization of PM resources. Utilization of resources determines power consumption of a PM. In fact, power consumed by a PM increases linearly with its utilization [3]. EARA uses relationship between power consumption of a PM and its utilization (Equation (15)) as a basis for energy consumption calculation.

$$P = P_{idle} + (P_{max} - P_{idle})U \tag{15}$$

where P_{idle} is the power consumption when PM is idle, P_{max} is the power consumption at 100 % utilization, and P is the power consumption at utilization $U \in (0, 1)$ of the PM. An idle PM consumes 70 % of the power consumed at full load [3]. Whereas, high utilization of PM resources causes performance degradation because tasks running on it do not get sufficient resources [3]. So, PM should not be operated at too low or very high utilization in order to improve its energy efficiency.

EARA assigns PM resources to VMs using ACO. The resource allocation using ACO is discussed in Section 4. Besides the optimal allocation and management of PM resources, DVFS is applied as recommended by Wu et al. in [27] to further reduce the energy consumption of a PM. In DVFS, energy is conserved by intentionally scaling down the performance of a PM as discussed in the next subsection.

3.2 Dynamic Voltage Frequency Scaling



a) Execution at maximum frequency b) Execution at different frequencies

Figure 3. Dynamic voltage frequency scaling

Modern processors support ACPI, and thus can be operated at different levels of voltage/frequency. This feature of modern processors has been extensively used in [37, 36, 34, 33, 32, 27] to reduce energy consumption as well as the heat dissipated by them. In this work, DVFS is used to execute a process/task using different combination of frequencies/voltages to conserve energy. Figure 3 shows the effect of executing a task using different combination of supported frequencies. Figure 3 a) depicts the case of executing a process/task at maximum frequency. The process may finish well in advance of its deadline time T. The completion of a task sometimes lowers the utilization of a PM below LGT. In such cases energy can be saved by either server consolidation or dynamic performance scaling of PM. Sometimes consolidation is not economically feasible due to constraints such as communication cost of migration, QoS violations due to interruption in service while consolidating, or unavailability of PM with sufficient free resources where the VM can be migrated. In such cases energy consumed by PMs can be saved by executing the process using different combinations of voltage/frequency [37, 36, 34, 33, 32, 27] as shown in Figure 3 b). In EARA, when the utilization of a PM drops below LGT and remains less than LGT for two consecutive monitoring periods, voltage/frequency of PM is adjusted to lower supported level to conserve energy. In case of low utilization, LNC sends current utilization value 'U' to DVFS module shown in Figure 4.

DVFS module calculates voltage/frequency level 'VF' that can complete the workload before user deadline. Voltage/frequency of PM is then adjusted to 'VF' to conserve energy. Energy is saved at the cost of slower process execution. Elongated process execution time does not affect user satisfaction because frequencies are selected in such a manner to complete the process before desired time deadline. The different frequencies are calculated to complete the task by deadline time T. The PMs in EARA are assumed to have to ACPI support and can be operated at any of the discrete h frequencies $f_1 < f_2 < \ldots < f_{h-1} < f_h$, supported by it. The key idea behind applying DVFS is to execute tasks using linear combination of supported frequencies. The minimization of power consumption is formulated as shown in Equation (16).

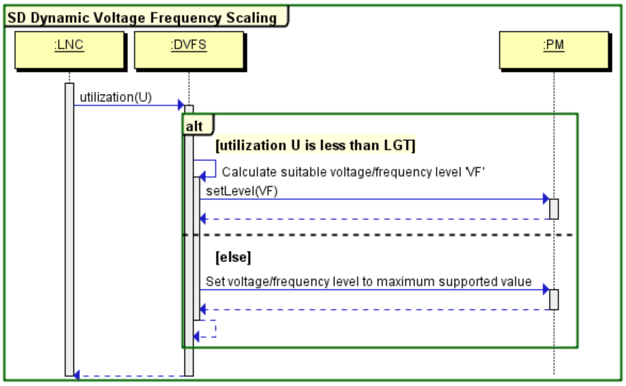

SD Dynamic Voltage Frequency Scaling
:LNC
:DVFS
:PM
utilization(U)
alt
[utilization U is less than LGT]
Calculate suitable voltage/frequency level 'VF'
setLevel(VF)
[else]
Set voltage/frequency level to maximum supported value

Figure 4. Sequence diagram for dynamic voltage frequency scaling

$$\min E_{ij} = \begin{cases} \sum_{l=1}^{h} t_l ACV^2 f_l, & \text{s.t.} \\ \sum_{l=1}^{h} t_l \leq T, t_l \geq 0, & \text{for } l = 1, 2, \dots, h \end{cases} \tag{16}$$

where h is number of supported frequencies, t_l is the time period for which the task is executed at frequency f_l.

3.3 Fitness Function

The main goal is to minimize total energy consumption, total execution time, and cost of execution by the efficient utilization of resources. We used weighted sum method to scalarize multiple objectives into a single objective. The weighted fitness function is calculated as shown in Equation (17).

$$F(N_t) = \xi(TEC_{min}) + \zeta(TET_{min}) + \gamma(COE) \tag{17}$$

where

$$TEC_{min} = \sum_{g=1}^{N_v} min(E_{ijg}), \qquad \forall i, j, \tag{18}$$

$$TET_{min} = \sum_{g=1}^{N_v} min(T_{ijg}), \qquad \forall i, j, \tag{19}$$

$0 \leq \xi, \zeta, \gamma < 1$, and $\xi + \zeta + \gamma = 1$. The value of weights ξ, ζ, and γ depends on the importance of each objective in the context of resource allocation problem. Mathematical model formulated for EARA is implemented using ACO metaheuristic as discussed in the next section.

4 ANT COLONY OPTIMIZATION BASED ENERGY AWARE RESOURCE ALLOCATION

Ant colony optimization is a metaheuristic optimization technique proposed by Dorigo in 1992 [9]. Ants secrete a chemical substance called pheromone while foraging. Pheromone gets deposited on the paths followed by ants. The amount of pheromone deposited on the path depends on the number of ants that followed the path. So a path that is used by large number of ants will have a higher quantity of the pheromone deposit. Initially, the ants choose random path while searching for food. But with time the difference in the quantity of pheromone deposited on the paths guides them to choose the path marked with a strong pheromone concentration. The larger amount of pheromone on a path attracts more ants to choose that path again, and finally all the ants converge to the single path. Pheromone evaporates with time, thus reducing the attractive strength of the path, causing ants to explore more paths to the food source. Pheromone evaporation has the advantage of avoiding the convergence to locally optimal solution.

Reasons behind choosing ACO for resource allocation are:

1. It can solve certain NP-hard problems in polynomial time.
2. It maintains a balance between acquired knowledge and exploring new solutions exploiting pheromone evaporation.
3. It gives near to optimal solution.
4. It performs distributed computation to avoid premature convergence.

Ant colony optimization has been applied to solve wide range of combinatorial optimization problems [7, 9, 10, 11, 29]. To solve a combinatorial optimization problem using ant colony optimization, an instance of the problem has to be mapped to a graph $G = (N, L)$, called construction graph. Node set N of the graph represents components of the problem instance and edge set L fully connects the components. Each edge (u, v) of the graph is associated with pheromone trail τ_{uv} and heuristic information η_{uv}. Heuristic information can be cost, distance, etc. that is associated with the edge. Each ant uses a pheromone trail and heuristic information, probabilistically, to construct its own solution of the problem instance. Once the solution is constructed the pheromone trail associated with every edge is updated to reflect evaporation, in order to enable ants to forget previously taken bad decision. The pheromone trail on each edge that belongs to the best solution is then updated.

In this paper, we have applied ACO at two levels for:

1. allocation of VM resources to jobs, and
2. allocation of PM resources to VMs.

Detailed description of the graph construction, pheromone and heuristic information, solution construction, pheromone evaporation, and pheromone trail update for allocation of VM resources to jobs and PM resource to VMs is as follows.

4.1 Allocation of VM Resources to Jobs

Each job of end users has some resource demands and QoS requirements. Each QoS parameter is associated with some value that indicates its priority over the others. The weights of QoS properties can be specified by three ways: absolute weighting, relative weighting, and arbitrary weighting [31]. In this work, we used relative weighting for QoS attributes. ACO metaheuristic for allocation of VM resources to jobs is explained below:

Construction Graph: The problem of VM resource allocation to jobs is mapped to construction graph $G_1 = (N_1, L_1)$. The node set N_1, consists of all VMs and jobs. Set L_1 of edges fully connects the nodes of the graph G_1. Each edge (a, g) of the graph G_1 is assigned pheromone value given by Equation (20).

$$\tau_{ag} = \frac{1}{\ell_a} \tag{20}$$

where a is unique identification number of a job, g is unique identification number of a VM, ℓ_a is the length of the job a. In general, length of job (cloudlet) is in millions of instructions. As inverse of job length is used as pheromone value so shorter jobs will be given preference over the longer ones. Heuristic information assigned to edge (a, g) is given by Equation (21).

$$\eta_{ag} = \prod_{x=1}^{X} (W_{ax})^{\alpha_x} \tag{21}$$

where X is number of QoS parameters associated with job a, W_{ax} is weight value of QoS parameter x, and α_x is control parameter for QoS attribute x of job a.

Solution Construction: Each ant is initially provided with the list of jobs to be mapped on VMs. For each job a that is mapped to VM g, variable y_{ag} is set to 1. Variable y_{ag} is used to keep record of jobs that have already been assigned. The probability that an ant k maps job a on VM g is:

$$\wp_{ag}^{k} = \begin{cases} \frac{(\tau_{ag})^{\alpha 1}(\eta_{ag})^{\beta 1}}{\sum_{t \in \mathcal{N}_g^k} (\tau_{tg})^{\alpha 1}(\eta_{tg})^{\beta 1}}, & \text{if } t \in \mathcal{N}_g^k, \\ 0, & \text{otherwise,} \end{cases} \tag{22}$$

where $\mathcal{N}_g^k$, consisting of all the jobs remaining to be mapped, is called feasible neighborhood of VM g. A job a which has maximum value of $\wp_{ag}^k$ is mapped to VM g. The probability of a job selection for mapping on a particular VM depends on the value of the pheromone trail and heuristic information of the associated edge. $\alpha 1$ is the parameter to control influence of pheromone trail, and $\beta 1$ is parameter to control the overall influence of weight values of QoS parameters. Both $\alpha 1$ and $\beta 1$ can have any value between 0 and 1, and their sum should be equal to one.

Pheromone Trail Evaporation: The pheromone deposited on all the arcs evaporates with time by a constant factor $0 \leq \rho_1 \leq 1$, called evaporation rate. Evaporation avoids unlimited accumulation of pheromone trails on the edges and enables ants to forget allocation decisions previously taken. Pheromone evaporation on all the edges of graph G_1 is realized by Equation (23).

$$\tau_{ag} = (1 - \rho_1)\tau_{ag}, \quad \forall (a, g) \in L_1. \tag{23}$$

Pheromone Trail Update: In order to reflect usage of an arc during solution construction, the pheromone trail on it is updated by an amount equal to inverse of the number of VMs used by an ant for the solution construction. The update makes pheromone concentration on some of the arcs stronger than the others. Strong pheromone concentration on an edge increases the probability of selection of the associated job. Pheromone update by ant k is realized as:

$$\tau_{ag} = \tau_{ag} + \sum_{k=1}^{N_a^1} \Delta\tau_{ag}^k, \quad \forall (a, g) \in S1^k \tag{24}$$

where N_a^1 is the number of ants, $\Delta\tau_{ag}^k$ is the amount of additional pheromone to be deposited on edge (a, g) which is traversed by ant k while constructing the allocation solution, and $S1^k$ is jobs to VMs mapping solution constructed by ant k. The amount of pheromone deposited on arc (a, g) by ant k is defined as follows:

$$\Delta\tau_{ag}^k = \begin{cases} \frac{1}{D1^k}, & (a, g) \in S1^k, \\ 0, & \text{otherwise}, \end{cases} \tag{25}$$

where $D1^k$ is number of VMs used for mapping of all jobs and calculated as length of solution $S1^k$.

4.2 Allocation of PM Resources to VM

Construction Graph: The resource allocation problem is mapped to construction graph $G_2 = (N_2, L_2)$. The node set N_2, consists of all VMs and PMs. L_2 is set of edges that fully connects nodes. Each edge (g, j) of the graph G_2 is associated with pheromone trail τ_{gj} and heuristic information η_{gj}, where g is the identification number of a VM and j is identification number of a PM. Pheromone trail associated with an edge(g, j) is given by Equation (26):

$$\tau_{gj} = \frac{1}{\frac{vm_g}{FM_j}} \tag{26}$$

where vm_g is memory requirements of VM g, and FM_j is available memory space of PM j which can be calculated for given g and j using Equation (27):

$$FM_j = hm_{ij} - \sum_g vm_{ijg} * x_{ijg}, \quad \forall i, j | x_{ijg} = 1. \tag{27}$$

Heuristic information assigned to the edge (g, j) for PM j of cluster i is given by Equation (28):

$$\eta_{gj} = \frac{1}{d_{ij}} \tag{28}$$

where d_{ij} is the distance between the frontend server and PM j of cluster i.

Solution Construction: Each ant is initially provided with the list of VMs to be deployed. x_{ijg} is set to 1 for each VM g that is assigned to PM j of cluster i. Variable x_{ijg} is used to keep record of VM that have already been assigned. The probability that an ant k deploys VM g on PM j of cluster i is:

$$\wp_{ijg}^k = \begin{cases} \frac{(\tau_{gj})^{\alpha 2}(\eta_{gj})^{\beta 2}}{\sum_{t \in \mathcal{N}_j^k} (\tau_{tj})^{\alpha 2}(\eta_{tj})^{\beta 2}}, & \text{if } t \in \mathcal{N}_j^k, \\ 0, & \text{otherwise,} \end{cases} \tag{29}$$

where $\mathcal{N}_j^k$ is the feasible neighborhood of PM j, comprising all those VMs which can still be deployed on it. The probability of choosing a PM j for VM g increases with the value of associated pheromone trail τ_{gj} and heuristic information η_{gj}. $\alpha 2$ and $\beta 2$ are the parameters to control the influence of pheromone trail and heuristic information, and can have any value between 0 and 1.

Pheromone Trail Evaporation: The pheromone deposited on all the arcs graph G_2 evaporates with time by a constant factor $0 \leq \rho_2 \leq 1$, is called the evaporation rate. Evaporation avoids unlimited accumulation of pheromone trails on the edges and enables ant to forget allocation decisions previously taken. Pheromone evaporation on all the edges of graph G_2 is realized by Equation (30).

$$\tau_{gj} = (1 - \rho_2)\tau_{gj}, \quad \forall (g, j) \in L_2. \tag{30}$$

Pheromone Trail Update: In order to reflect usage of an arc during solution construction, pheromone trail on it is updated by the amount equal to inverse of the length of solution path. The update makes pheromone concentration on some of the arcs stronger than the others. Strong pheromone concentration increases the probability of arc selection, which is exploited to optimize the utilization of PM. Pheromone update by ant k is realized as:

$$\tau_{gj}^k = \tau_{gj}^k + \sum_{k=1}^{N_a^2} \Delta\tau_{gj}^k, \quad \forall (g, j) \in S2^k \tag{31}$$

where N_a^2 is the number of ants, $\Delta\tau_{gj}^k$ is the amount of additional pheromone to be deposited on arc (g, j) which is traversed by ant k while constructing the solution, and $S2^k$ is solution constructed by ant k, consisting of VMs to PMs mapping. The amount of pheromone deposited on arc (g, j) by ant k is defined as follows:

$$\Delta\tau_{gj}^k = \begin{cases} \frac{1}{D2^k}, & (g, j) \in S2^k, \\ 0, & \text{otherwise}, \end{cases} \tag{32}$$

where $D2^k$ is computed as sum of lengths of all arcs belonging to solution $S2^k$.

Algorithm 1 EARA

```
1: procedure EARA($VM$)                                    ▷ set of VMs
2:     initialization
3:     while not Termination Condition do
4:         Allocation Solution Construction
5:         Pheromone Evaporation
6:         Pheromone Trail Update
7:     end while
8:     return VM to PM map                                 ▷ VM to PM mapping
9: end procedure
```

Algorithm 2 *Initialization*

```
1: procedure Initialization
2:     Create construction graph $G_2(N_2, L_2)$ of resource allocation problem
3:     Set VM = ID of nodes representing VMs in the construction graph $G_2(N_2, L_2)$
4:     Set PM = ID of nodes representing PMs in the construction graph $G_2(N_2, L_2)$
5:     for all $g$ in VM do
6:         for all $j$ in PM do
7:             Set $\tau_{gj} = \frac{1}{\frac{vm_g}{FM_j}}$
8:             i = getClusterID(j)                          ▷ get cluster ID of PM $j$
9:             Set $\eta_{gj} = \frac{1}{d_{ij}}$
10:        end for
11:    end for
12: end procedure
```

4.3 Algorithms for EARA

The pseudo code for energy aware resource allocation (EARA) using ACO shown in Algorithm 1, is composed of four phases namely; *initialization, allocation solution*

Algorithm 3 *Allocation Solution Construction*

```
1: procedure Resource Allocation(k)    ▷ Resource allocation solution construction for ant k
2:     input: Construction graph $G_2(N_2, L_2)$ of the resource allocation problem
3:     Set $S2^k$ = NULL    ▷ Initialize solution set $S2^k$ of ant k
4:     while not (Are all VMs Alloted?) do
5:         if Available(activePM) then
6:             Set j = ID of randomly selected PM from list of active PMs
7:             i = getClusterID(j)    ▷ get cluster ID of PM $j$
8:         else
9:             Set j = ID of randomly selected PM from list of newly added PMs
10:         ▷ Randomly selects PM representing dummy node of construction graph
11:             i = getClusterID(j)    ▷ get cluster ID of PM $j$
12:         end if
13:         Set $\mathcal{N}_j^k$ = NULL    ▷ feasible neighborhood of PM $j$
14:         Set VMIDs = IDs of VMs yet not assigned to any PM
15:         for all $g$ in { VMIDs } do
16:             if PM $j$ fulfills processing, memory and network bandwidth requirements of VM $g$ then
17:                 Add VM $g$ to $\mathcal{N}_j^k$ ▷ Add VM $g$ to feasible neighborhood of PM $j$
18:                 Evaluate $\wp_{ijg}^k = \frac{(\tau_{gj})^{\alpha 2}(\eta_{gj})^{\beta 2}}{\sum_{t \in \mathcal{N}_j^k}(\tau_{tj})^{\alpha 2}(\eta_{tj})^{\beta 2}}$,
19:             end if
20:         end for
21:         $VM_{id}^f$ = NULL    ▷ set the fittest VM to NULL
22:         $P^f = 0$    ▷ Probability of the fittest VM
23:         for all VM $g$ in $\mathcal{N}_j^k$ do
24:             if $\wp_{ijg}^k > P^f$ then
25:                 $VM_{id}^f = g$    ▷ set the fittest VM ID to $g$
26:                 $P^f = \wp_{ijg}^k$    ▷ Change probability of the fittest VM
27:             end if
28:         end for
29:         if $VM_{id}^f$ # NULL then
30:             Set $g = VM_{id}^f$    ▷ assign ID of the fittest VM to $g$
31:             Set $x_{ijg} = 1$    ▷ Assign VM $g$ to PM $j$ of Cluster $i$
32:             Add $x_{ijg}$ to $S2^k$    ▷ update allocation solution of ant $k$
33:         end if
34:     end while
35:     return ($S2^k$)
36: end procedure
```

Algorithm 4 *Pheromone Evaporation*

```
1: procedure PheromoneEvaporation
2:     Set VM = ID of nodes representing VMs in the construction graph $G_2(N_2, L_2)$
3:     Set PM = ID of nodes representing PMs in the construction graph $G_2(N_2, L_2)$
4:     for all $g$ in VM do
5:         for all $j$ in PM do
6:             $\tau_{gj} = (1 - \rho_2)\tau_{gj}$
7:         end for
8:     end for
9: end procedure
```

Algorithm 5 *Pheromone Trail Update*

```
1: procedure PheromoneUpdate(k)
2:     input: $k$                                   ▷ ant identifier
3:     $D2^k$ = Length($S2^k$)                      ▷ Calculate length of solution $S2^k$
4:     $\Delta\tau^k = \frac{1}{D2^k}$
5:     for all $s$ in $S2^k$ do                     ▷ for each element $s$ in solution $S2^k$
6:         $g$ = getVMID(s)                         ▷ get VM ID from solution element $s$
7:         $j$ = getPMID(s)                         ▷ get PM ID from solution element $s$
8:         $\tau_{gj} = \tau_{gj} + \Delta\tau^k$   ▷ update pheromone information on edge $(g, j)$
9:     end for
10: end procedure
```

construction, pheromone evaporation, and *pheromone trail update.* As discussed earlier ACO is applied at two levels. At first level, ACO is applied to allocate VM resources to jobs and at second level it is applied to allocate PM resources to VMs. We have discussed here the pseudo code for allocation of PM resources to VMs only. The given pseudo code can be easily modified for allocation of VM resources to jobs.

For allocation of PM resources to VMs, list of VMs is passed to the procedure EARA (Algorithm 1). In the *initialization* phase as shown in Algorithm 2, construction graph of the problem is created and every edge of the graph is assigned initial pheromone trail and heuristic information as discussed earlier. The function getClusterID(), in line number 8, is used to get cluster ID of a PM. Algorithm 3 outlines *allocation solution construction* phase. It probabilistically maps the VMs to the appropriate PMs. A PM is selected randomly and VMs from its feasible neighborhood are assigned to it one by one till UGT is observed. The process is repeated for each ant until all the VMs are mapped. When all the ants have finished assigning VMs to PMs, pheromone trail evaporation on all the edges is performed by Algorithm 4. Pheromone trail on every edge used by an ant for solution construction is then updated as shown in Algorithm 5.

5 PERFORMANCE EVALUATION AND COMPARATIVE ANALYSIS

The performance evaluation of EARA is performed on CloudSim. CloudSim is a framework for modeling and simulation of cloud computing infrastructure and services [5]. CloudSim simulation toolkit is used for implementation, testing, and validation because of large-scale nature of real cloud environment. For performance analysis, EARA is compared with existing resource allocation algorithms, i.e. FFD [30] and MGGA [28]. Five data centers are created with specification as shown in Table 1. In each data center, PMs complying with specifications, as shown in Table 2, are created. To conduct comparative analysis, four types of VMs, as shown in Table 3, are used. FFD, MGGA and EARA are rigorously tested with jobs having different QoS requirements.

Name	*PC*	*MC*	*SC*	*BC*	*Time Zone*
DC1	3	0.05	0.10	0.10	3.0
DC2	3.5	0.07	0.10	0.11	5.0
DC3	4	0.09	0.10	0.07	5.5
DC4	5	0.10	0.10	0.13	8.0
DC5	5.25	0.12	0.10	0.15	10.0

Name – data center name; PC – processing cost; MC – memory cost per MB; SC – storage cost per MB; BC – bandwidth cost; Time Zone – time zone of data center location

Table 1. Specification of data centers

PM Type	*CPU*	*Cores*	*RAM*	*Storage*	*BW*
1	1 000	4	8	2	10
2	1 500	8	16	2	10
3	2 000	12	32	2	10
4	3 000	20	64	4	10
5	5 000	36	64	4	10

CPU – processing speed in mips; Cores – number of processing cores; RAM – random access memory in GB; Storage – permanent storage capacity in TB; BW – network bandwidth in gbps

Table 2. Specification of physical machines

The aim of executing jobs with different QoS requirements is to test the effectiveness of EARA in terms of energy efficiency, number of PMs required, and quality of service. Performance of EARA is analyzed by varying number of jobs in every simulation run. Simulation is repeated 25 times and in each simulation run, parameters are set to a value from the range of values given in Table 4.

VM Type	*CPU*	*PEs*	*RAM*	*BW*
1	500	1	512	1
2	1 000	2	1 024	2
3	2 000	4	2 048	4
4	4 000	8	4 096	8

CPU – processing speed in mips; PEs – number of cores; RAM – random access memory in MB; BW – network bandwidth in gbps

Table 3. Specification of virtual machines

Number of jobs	200–1 200	Varied in every simulation run
Number of datastores	2–5	Stores instances of VMs
Number of ants	10–50	Construct allocation solution
Idle Time	10 min.	Time to switch PM to sleep mode
CPU_{UGT}	0.85	UGT for CPU utilization
MEM_{UGT}	0.85	UGT for memory utilization
BW_{UGT}	0.85	UGT for bandwidth utilization
CPU_{LGT}	0.30	LGT for CPU utilization
MEM_{LGT}	0.30	LGT for memory utilization
BW_{LGT}	0.30	LGT for bandwidth utilization

Table 4. Simulation parameters

5.1 Comparative Analysis

Figure 5 shows the comparison of the number of PMs used by FFD, MGGA, and EARA to fulfill the computational requirements of a given number of jobs. EARA outperforms both FFD and MGGA in terms of the number of PMs used to deploy a given number of jobs. On an average, EARA uses 11.36 % and 7.68 % lesser number of PMs than FFD and MGGA, respectively.

Figure 6 shows the comparison of total energy consumed by FFD, MGGA, and EARA. Total energy consumption of EARA is less than FFD and MGGA because it uses lesser number of PMs to deploy given number of jobs. It is experimentally established that EARA is 10.56 %, and 5.43 % more energy efficient than FFD and MGGA, respectively.

Figure 7 depicts the comparison of percentage resource utilization of PMs by FFD, MGGA, and EARA. In case of EARA, utilization of 85 % of PMs is between 41–80 %. In case of FFD and MGGA the utilization of more than one third of PMs is above 80 % which caused the performance degradation of user applications and resulted in creation of hot spots. Moreover, only 2 % of the PMs are there in EARA where the utilization is 0–20 % as compared to 5 % and 7 % in FFD and MGGA, respectively. Therefore, EARA is capable of managing the resources efficiently.

Figure 8 shows the comparison of the average number of VM migrations. The number of migrations in EARA is less than MGGA but more than FFD. In EARA, when the utilization of a PM falls below LGT value migration of VMs running over it is performed. Migration is performed for PMs consolidation so that some of the PMs can be switched to sleep mode to conserve energy. We observed approximately two migrations for 1200 jobs, such a small number of migrations does not impact the performance of the system. Moreover, EARA compensates the energy loss due to migrations by switching idle PMs to sleep mode.

Figure 9 shows the comparison of number of hot spots created by FFD, MGGA, and EARA as the number of jobs vary from 200 to 1200. We define a PM as a hot spot if its resource utilization is 100 %. Maximum number of hot spots are created by FFD methodology because it tries to utilize PM to its full capacity. However, EARA does not create any hot spot because it keeps resource utilization of PMs between LGT and UGT. Hot spots adversely affect the performance and reliability of PM. Moreover, creation of hot spots demands better cooling arrangements and also increases chances of hardware failure. Hence, EARA is more reliable and energy efficient.

Figure 10 shows the comparison of percentage workload of data centers, when $\alpha 2 = 0$. All the data centers have exactly same computing infrastructure but are at unequal distance from frontend server. Distance between a datacenter and frontend server is calculated from time zone of the datacenter. When $\alpha 2 = 0$, EARA gives weightage to distance while mapping VMs to PMs. As distance of DC1 is least so EARA distributes the VMs to PMs of DC1 first. Once DC1 resources are used upto UGT of their capacity, EARA starts assigning jobs to PMs of the datacenter whose distance is next to distance of DC1 and so on. EARA saves energy by deploying jobs/VMs over PMs which are nearer to frontend server because more energy is consumed to transmit jobs/VMs over longer distances. Moreover, deploying VM over nearer datacenter also improves response time.

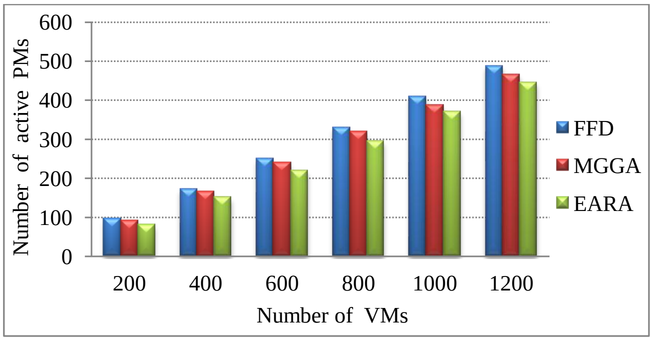


Figure 5. Comparison of number of PMs required by FFD, MGGA, and EARA

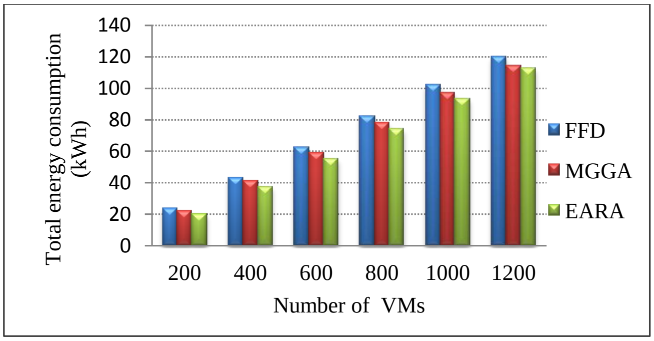


Figure 6. Comparison of total energy consumption by FFD, MGGA, and EARA

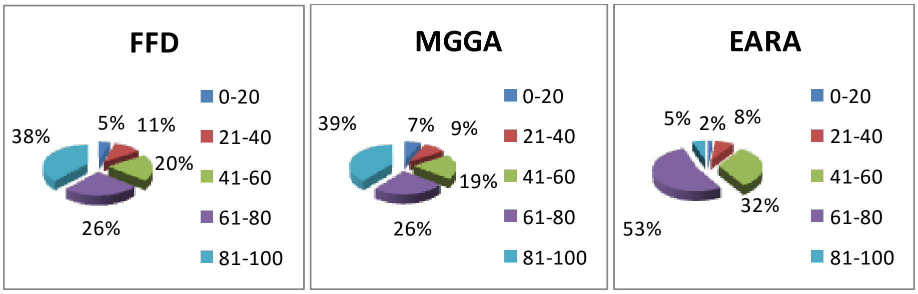


Figure 7. Comparison of PMs utilization in FFD, MGGA, and EARA

Figure 11 shows the comparison of percentage workload of data centers, when $\beta 2 = 0$. All the data centers are having five types of PMs with specification as per Table 2, and the data centers have equal number of specific type of PMs. In case of EARA, variance of the percentage workload of data centers is less than that of FFD and MGGA. This is due to the fact that EARA gives more weight to resource availability when $\beta 2$ is set to 0. As all the data centers have exactly the same computing infrastructure, so EARA distributes almost equal workload among them. This feature of EARA can be exploited for load balancing among datacenters.

Figure 12 shows the comparison of total energy consumption by the cloud computing infrastructure between EARA and EARA-DVFS. In case of EARA-DVFS, dynamic voltage frequency scaling is not applied. The result shows that EARA which is employing DVFS saves 8.15 % more energy than EARA-DVFS.

Figure 13 depicts the comparison of computational energy for FFD, MGGA, and EARA. It is the total energy consumed, measured in Watt hours (Wh), for finding suitable resources for all the jobs. This figure shows that the EARA consumes less energy in computation than MGGA but a little more than FFD. On an average, EARA consumes 0.42 % of total energy consumption if the allocation of resources is made efficiently. Therefore, EARA is better than FFD and MGGA as the compu-

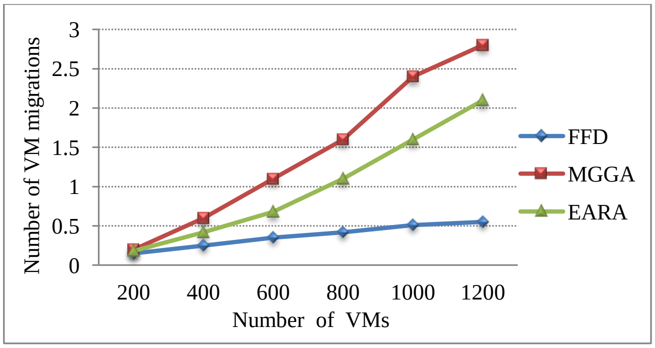


Figure 8. Comparison of number of VM migrations in FFD, MGGA, and EARA

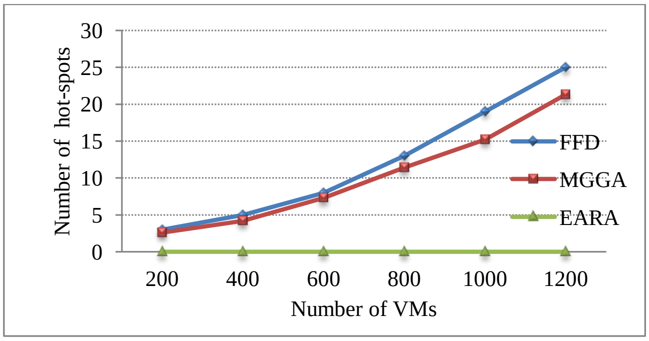


Figure 9. Comparison of number of hot spots in FFD, MGGA, and EARA

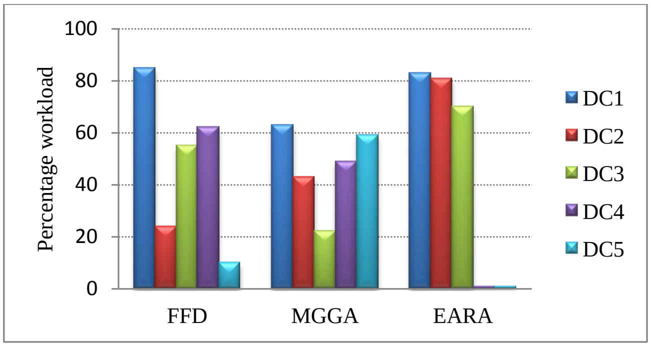


Figure 10. Comparison of percentage workload of data centers in FFD, MGGA, and EARA, when $\alpha 2 = 0$

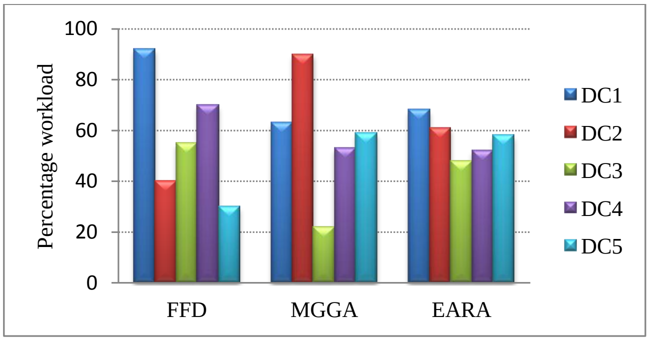


Figure 11. Comparison of percentage workload of data centers in FFD, MGGA, and EARA, when $\beta 2 = 0$

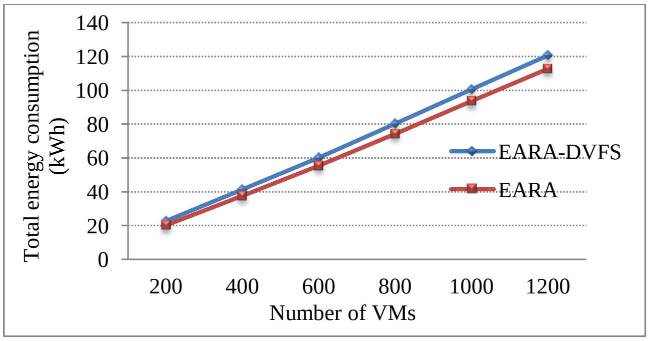


Figure 12. Comparison of total energy consumption between EARA and EARA-DVFS

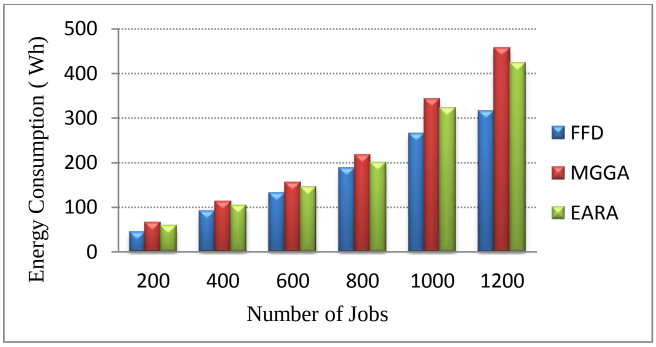


Figure 13. Comparison of computational energy consumption in FFD, MGGA, and EARA

tational energy consumption of 0.42 % is very small compared to the overall energy gain of 10.56 %.

6 CONCLUSION AND FUTURE WORK

In this paper, energy aware resource allocation methodology using the ant colony optimization has been proposed. ACO is applied at two levels for efficient allocation of resources. The first level ACO allocates VMs resources to jobs whereas the second level ACO allocates PMs resources to VMs. Resources are allocated to jobs on the basis of their QoS requirements. Server consolidation and dynamic performance scaling of PMs are employed to conserve energy. The proposed methodology is implemented in CloudSim and the results are compared FFD and MGGA resource allocation methods. It is experimentally established that the proposed EARA achieves up to 10.56 % saving in energy consumption through a better utilization of resources and desired QoS.

In future, EARA can be tested on OpenNebula based private cloud environment comprising water cooled CPU, and CPU with self steering frequency to confirm its capability of reducing the energy consumption. Furthermore, temperature aware resource allocation, and fault tolerant features such as check pointing and replication can be added to make EARA more robust and reliable.

Acknowledgement

This work is partially supported by the major project granted by the University Grants Commission, New Delhi, India. File No. 41-652/2012(SR).

Appendix A SYMBOLIC NOTATIONS USED IN EARA

Table A1: List of Symbols

Symbol	Definition
CL	Set of clusters
cl_i	i^{th} cluster
PM_i	Set of physical machines in i^{th} cluster
N_c	Number of clusters
N_v	Number of VMs
N_t	Number of tasks/jobs
N_a^1	Number of ants used in allocating jobs to VMs
N_a^2	Number of ants used in allocating VMs to PMs
H_i	Number of PMs in cluster i

Continued on next page

Table A1 – continued from previous page

Symbol	**Definition**
h_{ij}	PM j of cluster i
$hs_{ij}, hm_{ij}, hn_{ij}$	CPU speed, memory, and network bandwidth of PM j of cluster i
E_{ijg}	Energy consumed by VM g if executed on PM j of cluster i
E_{ij}	Energy consumption of PM j of cluster i
VM	Set of VMs
R_g	Requirement of VM g
r_{ij}^q	Resource requirement of task q executing on PM j of cluster i
t_{ij}^q	Execution time of task q on PM j of cluster i
p_{ij}^q	Power consumed for executing task q on PM j of cluster i
s_{ij}^q	CPU speed allocated to task q on PM j of cluster i
e_{ij}^q	Energy consumed for executing task q on PM j of cluster i
T_{ijg}	Total execution time of all tasks deployed on VM g
h	Number of voltage/frequency levels supported by PM
C_{ijg}	Total cost of running VM g on PM j of cluster i
x_{ijg}	1 if VM g is assigned to PM j of cluster i, 0 otherwise
EC_{ij}	Execution cost of VMs run on PM j of cluster i
v_g	VM g
vs_g, vm_g, vn_g	CPU speed, memory, and network bandwidth of VM g
n_g	Number of jobs allocated to VM g
G_1	Construction graph for jobs to VMs mapping
N_1	Set of nodes of construction graph G_1
L_1	Set of edges of construction graph G_1
α_1	Parameter to control influence of pheromone trail in G_1
β_1	Parameter to control influence of heuristic information in G_1
ρ_1	Pheromone evaporation rate for graph G_1
$\wp_{ag}^k$	Probability of mapping job a to VM g
W_{ax}	Weight value of QoS parameter x of job a
ℓ_a	Length of job a
X	Number of QoS parameters
y_{ag}	is 1 if job a is assigned to VM g, 0 otherwise
$\mathcal{N}_g^k$	Feasible neighborhood of VM g for ant k
$S1^k$	Ant k's jobs to VMs mapping solution
$D1^k$	Number of VMs used in solution k
G_2	Construction graph for VMs to PMs mapping
N_2	Set of nodes of construction graph G_2
L_2	Set of edges of construction graph G_2
$\tau_{u,v}$	Pheromone value associated with edge (u, v)
$\eta_{u,v}$	Heuristic information associated with edge (u, v)

Continued on next page

Table A1 – continued from previous page

Symbol	**Definition**
α_2	Parameter to control influence of pheromone trail in G_2
β_2	Parameter to control influence of heuristic information in G_2
ρ_2	Pheromone evaporation rate for graph G_2
$\mathcal{N}_j^k$	Feasible neighborhood of PM j for ant k
FM_j	Available memory space of PM j
$S2^k$	Ant k's VMs to PMs mapping solution
$D2^k$	Length of solution k
d_{ij}	Distance of PM j of cluster i from frontend server
i, j, g, k, q, k	Identifier for cluster, PM, VM, ant, task, and ant, respectively
ξ, ζ, γ	Weight values for TEC, TET, and COE, respectively
P_{idle}	Power consumption of idle PM
P_{max}	Power consumption of PM at 100 % utilization
U	Utilization of a PM
P	Power consumption of PM at U % utilization
CPU_{UGT}	Upper Green Threshold value for CPU
MEM_{UGT}	Upper Green Threshold value for Memory
BW_{UGT}	Upper Green Threshold value for Bandwidth
CPU_{LGT}	Lower Green Threshold value for CPU
MEM_{LGT}	Lower Green Threshold value for Memory
BW_{LGT}	Lower Green Threshold value for Bandwidth
$P_{dynamic}$	Dynamic power consumption
P_{static}	Static power consumption
A	Switching activity
C	Capacitance
V	Voltage
I_{leak}	Leaking current
f	Frequency

REFERENCES

[1] Armbrust, M.—Fox, A.—Griffith, R.—Joseph, A. D.—Katz, R. H.—Konwinski, A.—Lee, G.—Patterson, D. A.—Rabkin, A.—Stoica, I.—Zaharia, M.: Above the Clouds: A Berkeley View of Cloud Computing. Technical report No. UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009.

[2] Beloglazov, A.: Energy-Efficient Management of Virtual Machines in Data Centers for Cloud Computing. Ph.D. thesis, University of Melbourne, 2013.

[3] Beloglazov, A.—Abawajy, J.—Buyya, R.: Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing.

Future Generation Computer Systems, Vol. 28, 2012, No. 5, pp. 755–768, doi: 10.1016/j.future.2011.04.017.

[4] Beloglazov, A.—Buyya, R.: Managing Overloaded Hosts for Dynamic Consolidation of Virtual Machines in Cloud Data Centers under Quality of Service Constraints. IEEE Transactions on Parallel and Distributed Systems, Vol. 24, 2013, No. 7, pp. 1366—1379, doi: 10.1109/TPDS.2012.240.

[5] Calheiros, R. N.—Ranjan, R.—Beloglazov, A.—De Rose, C. A. F.—Buyya, R.: CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. Software: Practice and Experience (SPE), Vol. 41, 2011, No. 1, pp. 23–50.

[6] Castañé, G. G.—Núñez, A.—Llopis, P.—Carretero, J.: E-mc2: A Formal Framework for Energy Modeling in Cloud Computing. Simulation Modeling Practice and Theory, Vol. 39, 2013, pp. 56–75.

[7] Chen, H.—Cheng, A. M. K.—Kuo, Y.-W.: Assigning Real-Time Tasks to Heterogeneous Processors by Applying Ant Colony Optimization. Journal of Parallel and Distributed Computing, Vol. 71, 2011, No. 1, pp. 132–142, doi: 10.1016/j.jpdc.2010.09.011.

[8] Chen, J.-J.—Yang, C.-Y.—Kuo, T.-W.—Shih, C.-S.: Energy-Efficient Real-Time Task Scheduling in Multiprocessor DVS Systems. Asia and South Pacific Design Automation Conference (ASP-DAC '07), 2007, pp. 342–349, doi: 10.1109/ASP-DAC.2007.358009.

[9] Dorigo, M.—Stützle, T.: Ant Colony Optimization. The MIT Press, Cambridge, MA, USA, 2004.

[10] Feller, E.—Rilling, L.—Morin, C.: Energy-Aware Ant Colony Based Workload Placement in Clouds. 12th IEEE/ACM International Conference on Grid Computing (GRID), 2011, pp. 26–33, doi: 10.1109/Grid.2011.13.

[11] Gao, Y.—Guan, H.—Qi, Z.—Hou, Y.—Liu, L.: A Multi-Objective Ant Colony System Algorithm for Virtual Machine Placement in Cloud Computing. Journal of Computer and System Sciences, Vol. 79, 2013, pp. 1230–1242, doi: 10.1016/j.jcss.2013.02.004.

[12] Garg, S. K.—Yeo, C. S.—Buyya, R.: Green Cloud Framework for Improving Carbon Efficiency of Clouds. Euro-Par 2011 Parallel Processing. Lecture Notes in Computer Science, Vol. 6852, 2011, pp. 491–502, doi: 10.1007/978-3-642-23400-2_45.

[13] Hsu, C.-H.—Chen, S.-C.—Lee, C.-C.—Chang, H.-Y.—Lai, K.-C.—Li, K.-C.—Rong, C.: Energy-Aware Task Consolidation Technique for Cloud Computing. IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom), 2011, pp. 115–121, doi: 10.1109/CloudCom.2011.25.

[14] Huang, C. J.—Guan, C.-T.—Chen, H.-M.—Wang, Y.-W.—Chang, S.-C.—Li, C.-Y.—Weng, C.-H.: An Adaptive Resource Management Scheme in Cloud Computing. Engineering Applications of Artificial Intelligence, Vol. 26, 2013, No. 1, pp. 382–389.

[15] Kaplan, J. M.—Forrest, W.—Kindler, N.: Revolutionizing Data Center Energy Efficiency. Technical report, McKinsy & Company, 2008.

[16] Kim, N.—Cho, J.—Seo, E.: Energy-Credit Scheduler: An Energy-Aware Virtual Machine Scheduler for Cloud Systems. Future Generation Computer Systems, Vol. 32, 2014, pp. 128–137, doi: 10.1016/j.future.2012.05.019.

[17] Kinger, S.—Kumar, R.—Sharma, A.: Prediction Based Proactive Thermal Virtual Machine Scheduling in Green Clouds. The Scientific World Journal, Vol. 2014, 2014, Art. No. 208983, pp. 92–103, doi: 10.1155/2014/208983.

[18] Koomey, J.: Growth in Data Center Electricity Use 2005 to 2010. 2014, `http://www.analyticspress.com/datacenters.html`.

[19] Lee, H. M.—Jeong, Y.-S.—Jang, H. J.: Performance Analysis Based Resource Allocation for Green Cloud Computing. The Journal of Supercomputing, Vol. 69, 2014, No. 3, pp. 1013–1026.

[20] Mell, P.—Grance, T.: The NIST Definition of Cloud Computing, 2011.

[21] Nathani, A.—Chaudhary, S.—Somani, G.: Policy Based Resource Allocation in IaaS Cloud. Future Generation Computer Systems, Vol. 28, 2012, No. 1, pp. 94–103, doi: 10.1016/j.future.2011.05.016.

[22] Nathuji, R.—Schwan, K.: VirtualPower: Coordinated Power Management in Virtualized Enterprise Systems. ACM SIGOPS Operating Systems Review, Vol. 41, 2007, No. 6, pp. 265–278, doi: 10.1145/1294261.1294287.

[23] Quarati, A.—Clematis, A.—Galizia, A.—D'Agostino, D.: Hybrid Clouds Brokering: Business Opportunities, QoS and Energy-Saving Issues. Simulation Modeling Practice and Theory, Vol. 39, 2013, pp. 121–134, doi: 10.1016/j.simpat.2013.01.004.

[24] Raycroft, P.—Jansen, R.—Jarus, M.—Brenner, P. R.: Performance Bounded Energy Efficient Virtual Machine Allocation in the Global Cloud. Sustainable Computing: Informatics and Systems, Vol. 4, 2014, pp. 1–9.

[25] Takeda, S.—Takemura, T.: A Rank-Based VM Consolidation Method for Power Saving in Datacenters. IPSJ Transactions on Advanced Computing Systems, Vol. 3, 2010, No. 2, pp. 138–146, doi: 10.2197/ipsjtrans.3.88.

[26] Vogels, W.: Beyond Server Consolidation. Queue, Vol. 6, 2008, No. 1, pp. 20–26, doi: 10.1145/1348583.1348590, doi: 10.1145/1348583.1348590.

[27] Wu, C.-M.—Chang, R.-S.—Chan, H.-Y.: A Green Energy-Efficient Scheduling Algorithm Using the DVFS Technique for Cloud Datacenters. Future Generation Computer Systems, Vol. 37, 2014, pp. 141–147, doi: 10.1016/j.future.2013.06.009.

[28] Xu, J.—Fortes, J. A. B.: Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments. IEEE/ACM International Conference on Green Computing and Communications (GreenCom), 2010, pp. 179–188, doi: 10.1109/GreenCom-CPSCom.2010.137.

[29] Yin, P.-Y.—Wang, J.-Y.: Ant Colony Optimization for the Nonlinear Resource Allocation Problem. Applied Mathematics and Computation, Vol. 174, 2006, No. 2, pp. 1438–1453.

[30] Yue, M.: A Simple Proof of the Inequality FFD (L) $\leq$ 11/9 OPT (L) + 1, $\forall$L for the FFD Bin-Packing Algorithm. Acta Mathematicae Applicatae Sinica, Vol. 7, 1991, No. 4, pp. 321–331.

[31] Tran, V. X.—Tsuji, H.—Masuda, R.: A New QoS Ontology and Its QoS-Based Ranking Algorithm for Web Services. Simulation Modelling Practice and Theory, Vol. 17, 2009, No. 8, pp. 1378–1398.

[32] Wang, P.—Qi, Y.—Liu, X.: Power-Aware Optimization for Heterogeneous Multi-Tier Clusters. Journal of Parallel and Distributed Computing, Vol. 74, 2014, No. 1, pp. 2005–2015.

[33] Guérout, T.—Monteil, T.—Da Costa, G.—Calheiros, R. N.—Buyya, R.—Alexandru, M.: Energy-Aware Simulation with DVFS. Simulation Modelling Practice and Theory, Vol. 39, 2013, pp. 76–91, doi: 10.1016/j.simpat.2013.04.007.

[34] Zhai, B.—Blaauw, D.—Sylvester, D.—Flautner, K.: Theoretical and Practical Limits of Dynamic Voltage Scaling. Proceedings of the 41st Design Automation Conference, 2004, pp. 868–873, doi: 10.1145/996566.996798.

[35] Data Flow Diagram. 2015, `https://en.wikipedia.org/wiki/Data_flow_diagram`. Accessed: January 5, 2015.

[36] Marzolla, M.—Mirandola, R.: Dynamic Power Management for QOS-Aware Applications. Sustainable Computing: Informatics and Systems, Vol. 3, 2013, No. 4, pp. 231–248.

[37] Bertsekas, D. P.: Chapter 4 – Exact Penalty Methods and Lagrangian Methods. In: Bertsekas, D. P. (Ed.): Constrained Optimization and Lagrange Multiplier Methods. Academic Press, 1982, pp. 179–301, doi: 10.1016/B978-0-12-093480-5.50008-8.

Ashok Kumar received his M.Sc. degree in information technology from Punjab Technical University, Jalandhar. Currently he is pursuing his doctoral degree in cloud computing from Thapar University, Patiala. His research interests include cloud computing, internet of things and fog computing. He has five research publications in reputed journals and conferences.

Rajesh Kumar is currently working as Professor in the Computer Science and Engineering Department, Thapar University, Patiala. He received his M.Sc., M.Phil. and Ph.D. degrees from IIT Roorkee. He has more than 21 years of UG & PG teaching and research experience. He wrote over 101 research papers for various international and national journals and conferences. He has, so far, guided 10 Ph.D. and 23 M.E./M.Sc. theses. His current areas of research interests include FANETs, resource scheduling and fault tolerance in clouds.

Anju Sharma is working as Assistant Professor in the Computer Science and Engineering Department, MRSPTU, Bathinda. Her research interests include smart grid computing, cloud computing, IoT and fog computing. She has varied numbers of publications in international journals and conferences of repute. She is Senior Member of International Association of Computer Science and Information Technology (IACSIT) and professional member of ACM India, IEEE. She is an active member (TCM and reviewer) of varied conferences.