The CSI Journal on
Computer Science and Engineering
Vol. 10, No. 2 & 4 (b), 2012
Pages 28-38
Regular Paper

A Profit-Aware Allocation of High Performance Computing Applications on Distributed Cloud Data Centers with Environmental Considerations

Hamid Reza Faragardi[1] Aboozar Rajabi[2] Thomas Nolte[1]
Amir Hossein Heidarizadeh[3]

[1]School of Innovative Design and Engineering, Malardalen University, Sweden
[2]School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Iran
[3]School of Applied Mathematics, Malardalen University, Sweden

Abstract

A Set of Geographically Distributed Cloud data centers (SGDC) is a promising platform to run a large number of High Performance Computing Applications (HPCAs) in a cost-efficient manner. Energy consumption is a key factor affecting the profit of a cloud provider. In a SGDC, as the data centers are located in different corners of the world, the cost of energy consumption and the amount of CO_2 emission significantly vary among the data centers. Therefore, in such systems not only a proper allocation of HPCAs results in CO_2 emission reduction, but it also causes a substantial increase of the provider's profit. Furthermore, CO_2 emission reduction mitigates the destructive environmental impacts. In this paper, the problem of allocation of a set of HPCAs on a SGDC is discussed where a two-level allocation framework is introduced to deal with the problem. The proposed framework is able to reach a good compromise between CO_2 emission and the providers' profit subject to satisfy HPCAs deadlines and memory constraints. Simulation results based on a real intensive workload demonstrate that the proposed framework enhances the CO_2 emission by 17% and the provider's profit by 9% in average.

Keywords: Cloud Computing, Data Center, Energy-Aware Allocation, CO_2 Emission, Multi-Objective Optimization, Live Migration.

1. Introduction

Recently, a strong trend has emerged to utilize a Set of Geographically Distributed Cloud data centers (SGDC) to run a large number of High Performance Computing Applications (HPCAs). There are several advantages behind this trend such as fault tolerant benefits, high scalability, energy saving, etc. Cloud providers promise to prepare such infrastructure on demand with a minimum investment for customers through the cloud data centers. The data centers are geographically distributed over the world to support users in any corner of the world. In such a large-scale system, energy-efficient computing is a tremendous challenge [2].

Cloud data centers consume a large amount of energy which directly impacts on the cost of services and thereby on the cloud providers' profit as well. Based on a set of recent estimations, energy consumption has a substantial contribution to the total operational cost of data centers [3]. Higher energy consumption would make the services more expensive and thus in order to provide a service with a reasonable cost, system developers have to strive to design the system with the minimum possible energy consumption. In addition, an increase of the energy consumption will result in destructive impacts on the environment along with more greenhouse gas emissions [4]. In 2007, Gartner estimated that the Information and Communication Technologies (ICT)

industry generates about 2%of the total global CO_2 emissions, which is equal to the aviation industry [5].

Our goal in this paper is to find an optimal allocation of a set of HPCAs onto the servers of a SGDC to optimize both the provider's profit and the amount of CO_2 emission. It should be noted that minimizing energy consumption potentially improves both the amount of CO_2 emission and the cloud provider's profit. However, the problem in this article is different with the problem of minimizing energy consumption in some senses. As the cost of energy and the Coefficient of Performance (COP) are not necessarily identical in different data centers, minimizing energy consumption solely is not sufficient to optimize the provider profit. We also should attempt to allocate the major part of workload on the data centers which have a lower energy cost along with a higher COP. This is the case also for the amount of CO_2 emission. Since different places generate electricity by utilizing different techniques (such as fossil fuels, wind energy, water and so on), data centers usually have various carbon emission rates. Hence, to minimize the amount of CO_2 emission the HPCAs should be allocated to the data centers which exploit a green source of energy like wind or solar energy.

In order to achieve a better profit for cloud providers, energy consumption should be reduced. This energy reduction potentially results in a lower CO2 emission however, as other factors are also relevant to the amount of CO2 emission (such as the source of generating electrical energy), maximizing cloud providers' profit is not necessarily equal to minimizing CO2 emission. Therefore, the desirable solution is to find a good compromise between the profit and carbon emission. Moreover, in order to support quality of service (QoS) requirements, the proposed allocation frame work must be able to meet all HPCAs' deadlines. In addition to timing requirements, memory and the number of processors required to execute an HPCA are other principal constraints which should be addressed by the allocation framework. In such systems, each HPCA is usually assigned to a single Virtual Machine (VM) which provides the possibility of supporting live VM migration during running an HPCA from one server to another. Accordingly, the VM should be able to cover all requirements of the corresponding HPCA. For example, if a HPCA requires at least N processing cores to execute, then the VM to which the HPCA is assigned will demand cores. Therefore, from the allocation perspective, an HPCA and its corresponding VM can be applied interchangeably.

In this paper, the problem is formulated and then in order to cope with the problem, a two-level allocation frame work is introduced. The framework is an online approach comprising a two-level solution where the first level allocates the VMs corresponding to HPCAs onto the data centers (called data center-level allocator) while the second level plays the role of the local scheduler on each data center (called scheduler). The allocator is developed based on a powerful meta-heuristic approach known as the Imperialist Competitive Algorithm (ICA) [6]. Furthermore, to increase the convergence speed of ICA, a fast local search inspired from the Taboo search is applied. As the scheduler within each data center, a greedy heuristic algorithm is implemented called Highest Execution time-Lowest Power consumption (HELP). The scheduler and the allocator cooperate with each other in such a way that all the mentioned constraints are satisfied while achieves a right compromise between the profit and the amount of CO_2 emission. In addition, another algorithm is suggested to handle the live migration of VMs across the servers called migration handler. Integrating live VM migration to the proposed framework leads to a higher efficiency however, on the other hand, it increases the complexity of the problem.

The main contributions of this article are:

1. Proposing an online allocation frame work which works based on a two-level allocation architecture.
2. Contemplating real-time HPCAs and various system constraints.
3. Covering the live migration of VMs among the servers to address the problem more efficiently.

The remainder of this paper is organized as follows. A short brief of the related works are reviewed in Section 2. The problem definition is stated in Section 3. After describing the problem and underlying models, Section 4 discusses the proposed solution. Section 5 evaluates the solution approach and presents the obtained results. Finally, concluding remarks and future works are presented in Section 6.

2. Related Works

The energy consumption of data centers has been recently considered in several works. However, most of these approaches focus on scheduling of applications within one data center [18] [23]. On the other hand, CO_2 emission has been ignored in a wide range of previous works. There are some studies in Grids which investigate energy efficiency of resources in multiple locations, similar to our work. Orgerie et al. proposed a prediction algorithm that consolidate workloads on a portion of CPUs and turn off unused CPUs [7]. Patel et al. investigated allocating Grid workload at different data centers considering temperature [8]. The main goal of their work is reducing the temperature based on the energy efficiency at different data centers however, they ignored the CO_2 emission.

In the scope of cloud computing, a similar problem has discussed recently. Garg et al. have considered reducing the CO_2 footprint of cloud data centers by presenting a carbon-aware green cloud architecture [9]. Some heuristics are proposed for scheduling of high performance computing workloads in several data centers with both carbon and profit considerations [10]. They considered the provider's profit and CO_2 emission as the scheduling objectives. Although the proposed heuristics strive to find a good tradeoff between objectives, this approach can only optimize one goal at time. Kessaci et al. [11] have solved the same problem by applying a multi-objective genetic algorithm that optimizes the energy consumption, CO_2 emissions and the generated profit of a geographically distributed cloud computing infrastructure. They also propose a greedy heuristic that aims to maximize the number of scheduled applications in order to compare it with the multi-objective genetic algorithm. The mentioned studies [9, 10, 11] consider a homogeneous data centers where all the servers within a data center are identical whereas, it is not the case in several practical data centers and htus we address the problem for heterogeneous datacenters.

Khosravi et al. [20] have taken Power Usage Effectiveness (PUE) into account besides CO_2 emission for placing VMs in the data centers. They considered the VM placement problem as a bin packing problem and proposed a best-fit heuristic as an online allocation algorithm. However, they ignored Cloud providers' profit and VM migration in their solution framework. Similar problem was solved by [21] where they modeled the problem just like [20] while they proposed another heuristic algorithm utilizing VM migration to save more energy and reach to a lower carbon emission. Their solution divides the problem into two phases. i) Initial placement of VM to a suitable host from the datacenter having minimum carbon footprint rate among all the available hosts from different distributed data centers. ii) Optimization of current VM allocation inside every datacenter. Dividing the problem into two independent phases causes to find local optima, because, these two phases are inherently dependent to each other. Hence, to reach a global optimum they have to considered together as one optimization problem. Jing et al. [22] have proposed a meta-heuristic solution but they did not consider VM migration and also their solution is not suitable to apply as an online allocation algorithm due to its long execution time.

This work is different from the mentioned works in some senses. The first major difference is that we address the whole problem as one optimization problem which dramatically increases our chance to find a global solution. As the second substantial difference, we utilize the power of a problem-based heuristic algorithm with a meta-heuristic algorithm while the possibility of live migration of VMs is also taken into consideration. Indeed, the article can be considered as an extension idea discussed in [19].

3. Problem Definition

3.1. System Model

The system is a set *C* of *c* data centers which compose a SGDC. Execution price, CO_2 emission rate, electricity price and COP are considered as energy efficiency factors. These factors vary across different data centers depending on their locations, architectural designs and management systems. In addition, the number and heterogeneity of servers within the datacenters directly impact on the complexity of the problem. Each datacenter is a set *P* of *p* heterogeneous servers where the *i*th server has a specific number of cores denoted by γ_i along with the limited amount of memory indicated by mem_i.

The presumed service delivery model in this work is the Infrastructure-as-a-Service (IaaS). The service presented by the cloud provider is the offering of an infrastructure to run the clients' HPC applications. The workload is represented by a set a comprising of *N* HPCAs (VMs). Each HPCA has adeadline that must be met. A user submits his requirement for an HPCA$a_i$in the form of a tuple (d_{a_i}, n_{a_i},$e_{a_i p_j}$, m_{a_i}), where $d_{a_i}$is the relative deadline to complete a_i, $n_{a_i}$is the number of CPUs needed to executea_i,$e_{a_i p_j}$ is one of the elements of the vector *e* that represents the execution time of $a_i$on server $p_j$when that server is operating at the maximum frequency, $m_{a_i}$is the memory required by a_i.Moreover, VM_i implies the VM to which a_i is assigned.

3.2. Energy Model

The energy consumption of a data center is related to IT equipment such as servers and network, or other auxiliary equipment like cooling equipment and lightning. The lightning portion could be neglected because of its little impact on the total energy consumption of a data center [11]. As the energy consumption of the servers and cooling system are accountable for the significant portion of a datacenter's energy consumption, we ignore the network energy in this work, and it can be considered as part of our future work.

Due to the heterogeneity of servers within the data centers, the energy consumption of each application depends on both the data center to which the application is assigned and the server within the data center to which the application is allocated. Therefore, the server which is in charge of running the application should be known in order to calculate the total energy consumption by a set of applications. It should be mentioned that in this work, only energy usage of the CPU is considered as the energy consumption of a server. In other words, the energy consumption by other components (e.g., memory and disk) is ignored because CPU is the dominant part in terms of energy consumption when running CPU-intensive workloads. Hence, the power consumption in each server is derived from the power model in Complementary Metal-Oxide Semiconductor (CMOS) logic circuits which is defined by

$$\mathrm{P} = \mathrm{A'C'V^2f} + \mathrm{I_{leak}V} + \mathrm{P_{short}} \quad (1)$$

where A' is the number of switches per clock cycle, $C'$is the total capacitance load, V is the supply voltage, f is the frequency, I_{leak} is the leakage current and P_{short} is the power dissipation resulting from switching between a voltage to another. As A' and C' are constant, we replace their product by α. Moreover, the second part of the equation represents the static consumption, let it be β. In CMOS processors the voltage can be expressed as a linear function of the frequency and thus,$V^2 f$ can be replaced by f^3. Therefore, the energy consumption of the computing equipment for execution of $a_i$is computed by

$$\mathrm{E^{CE}_{a_i c_k}(X)} = \sum_{\mathrm{j=0}}^{\mathrm{p}} (\alpha_{\mathrm{p_j}} \mathrm{f^3_{p_j}} + \beta_{\mathrm{p_j}}) \times \mathrm{e_{a_i p_j}}\ \mathrm{x_{ij}} \quad (2)$$

where *X* is the assignment matrix in whichx_{ij} is equal to one if a_i is assigned to the *j*th server and otherwise, it is zero. The energy consumed by the cooling equipment is directly related to the location of the data center due to variance of temperature. The COP parameter could be used to compare the energy consumption of the cooling system [12, 13]. The COP indicates the ratio of energy consumed for execution of the HPCA to the amount of energy which is required for cooling the system. Indeed, COP represents the efficiency of the cooling system. Although the COP varies over time, we suppose that it is constant during one scheduling period. It is worth noting that the scheduling period is time interval between two subsequent invocations of allocation algorithm. The energy consumption of the cooling equipment of the data center c_j for the *i*th application, $E^{CS}_{a_i c_j}$, is defined by

$$\mathrm{E^{CS}_{a_i c_j}} = \mathrm{E^{CE}_{a_i c_k}} / \mathrm{COP_{c_j}} \quad (3)$$

According to Eq. 2 and 3, total energy consumed by applicationa_i executing on data center$c_j$is computed by

$$\mathrm{E}_{\mathrm{a_i c_j}}^{\mathrm{total}} = \mathrm{E}_{\mathrm{a_i c_j}}^{\mathrm{CS}} + \mathrm{E}_{\mathrm{a_i c_j}}^{\mathrm{CE}} = (1 + 1/\mathrm{COP}_{\mathrm{c_j}}) \times \mathrm{E}_{\mathrm{a_i c_j}}^{\mathrm{CE}} \quad (4)$$

3.3. CO_2 Emission Model

The amount of CO_2 emissions of the data center $c_i$is related to a coefficient. This coefficient, $r_{c_i}^{CO_2}$, is determined based on the method that the required electricity of $c_i$is generated. As we know, there are different ways for generating electricity such as using fossil fuels like oil and natural gas or using renewable resources like water, solar and wind power. The renewable resources are green and will make less destructive impacts on the environment.

Due to the diverse methods of generating electricity in various places, the value of $r_{c_i}^{CO_2}$ is different for eachcloud data center. The CO_2 emission due to the execution of applicationa_i on the data center $c_j$is computed by

$$\mathrm{CO_2E}_{\mathrm{a_i c_j}} = \mathrm{r}_{\mathrm{c_j}}^{\mathrm{CO_2}} \times \mathrm{E}_{\mathrm{a_i c_j}}^{\mathrm{total}} \quad (5)$$

where $r_{c_j}^{CO_2}$ is the CO_2 emission rate ofc_j. As a result, the total CO_2 emission incurred by the execution of all HPCAsis defined by

$$\mathrm{TCO_2E\,(X)} = \sum_{\mathrm{i=1}}^{\mathrm{N}} \sum_{\mathrm{j=1}}^{\mathrm{c}} \mathrm{CO_2E}_{\mathrm{a_i c_j}}(\mathrm{X}) \quad (6)$$

3.4. Profit Model

Profit is equal to income minus cost. In this paper, we define the income as the price that should be paid by the user. Also, the cost is the price which is incurred by electricity usage. The achieved profit due to the execution of application a_i in the data center c_i is computed by

$$\mathrm{Prof}_{\mathrm{a_i c_j}} = \mathrm{n}_{\mathrm{a_i}} \times \mathrm{e}_{\mathrm{a_i c_j}} \times \mathrm{p}_{\mathrm{a_i}}^{\mathrm{c}} - \mathrm{p}_{\mathrm{c_j}}^{\mathrm{e}} \times \mathrm{E}_{\mathrm{a_i c_j}}^{\mathrm{total}} \quad (7)$$

where $e_{a_i c_j}$ is the average execution time of a_i on c_j, $p_{a_i}^c$ is the static price of a_i, $p_{c_j}^e$ is electricity price of the area in which the c_j is located and $E_{a_i c_j}^{total}$ is the total energy consumption of a_i on c_j. Therefore, the total profit can be computed as $TProf(X)$ by

$$\mathrm{TProf\,(X)} = \sum_{\mathrm{i=0}}^{\mathrm{a}} \sum_{\mathrm{j=0}}^{\mathrm{c}} \mathrm{Prof}_{\mathrm{a_i c_j}}(\mathrm{X}) \quad (8)$$

3.5. Optimization Problem

The problem can be defined as an optimization problem as follows:

$$\begin{aligned} \text{Min} \quad & \theta.\mathrm{TCO_2E\,(X)} + \rho/\,\mathrm{TProf(X)}, \quad (9) \\ \text{Subject to} \quad & \sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{m}_{\mathrm{a_i}} \mathrm{x}_{\mathrm{ij}} \le \mathrm{mem_j} \, \mathrm{j} \in \text{all servers} \\ & \sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{n}_{\mathrm{a_i}} \mathrm{x}_{\mathrm{ij}} \le \gamma_{\mathrm{j}} \, \mathrm{j} \in \text{all servers} \\ & \mathrm{start}_{\mathrm{a_i p_j}} + \mathrm{e}_{\mathrm{a_i p_j}} \le \mathrm{d_i} \, \mathrm{i} \in \text{all HPCAs} \\ & \sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{x}_{\mathrm{ij}} = 1 \mathrm{j} \in \text{all servers} \end{aligned}$$

where coefficients θ and ρ are real numbers in the range of zero and one used to show the importance of each goal.Decision-maker can tune the value of them with respect to importance of each goal and if they have the same importance they should be set equal to 0.5. In addition, $start_{a_i p_j}$shows the start time of the *i*th application on the *j*th server if it is allocated to this server, otherwise, it is equal to zero.

4. Proposed Solution

In this section, an online allocation framework is suggested to cope with the formulated problem. The architecture of the proposed allocation framework consists of a two-level solution in which the first level deals with the allocation of VMs onto the data centers and the latter tackles with the allocation of VMs among the servers within a data center. It should be mentioned that in order to find an acceptable solution for the problem both solution levels collaborate together to find a global solution for the problem, in other words, the solution framework is working according to a feedback-based approach. In the following, the architecture is described in details and it is demonstrated how this frame work can be applied to address the problem.

4.1. Architecture

Data center-level allocation (Allocator): The allocator is located at the high level to partition a set of HPC Asamong the available data centers in a SGDC. In this level, the HPCAs (actually, their corresponding VMs) are mapped to the data centers in such a way that a right compromise between the profits and CO_2 emission can be achieved. Although this decision is made at the data center-level, the allocator should be aware of which server(s) of the data centeris supposed to execute the VMs. However, in most of the previous studies, since only homogenous data centers are taken into account, the allocator does not need to be aware of the allocation within the data centers. Because, if the servers of a data center are the same then, calculating the energy consumption of a VM is not dependent to the type of server running the VM.

Nevertheless, we introduce an intelligent architecture which is able to separate these two allocation levels even for heterogeneous data centers and provides us a two-level allocation framework. The data center-level allocator is inspired by the ICA algorithm intensified by a fast local search. As we mentioned above, ICA is responsible to find an appropriate mapping of VMs among the data centers. Each mapping is represented by a vector of *N* elements, and each element is an integer value between one and *c*. The vector is called *SM*. Figure 2 shows an illustrative example for a mapping of VMs. The third element of this example is two, which means that the third VM (corresponding to the third HPCA) is mapped to the second data center.

Furthermore, this representation causes satisfaction of the no redundancy constraint in the sense that each HPCA should be mapped to no more than one data center. Section 4.2 describes the ICA in more details.

a_1	a_2	a_3		a_N
1	1	2	…	1

Figure 1. Representation for mapping of services to the data centers

Server-level allocation (Scheduler): Each data center has a local scheduler. The submitted VMs to this data center are scheduled by the corresponding scheduler. The scheduler at this level aims to find an allocation which can meet the minimum number of processing cores to execute a VM along with the memory and deadline constraints while at same time it attempts to minimize the energy consumption. Decreasing energy consumption in a data center potentially leads to mitigation of CO_2 emission and increasing the cloud provider profit. It should be mentioned that the scheduler may not be able to find a feasible allocation for all the VMs dedicated to this datacenter. In other words, if all the VMs mapped to this data center are allocated to its servers, then some of them may violate at least one of the following constraints:

1. Missing their deadlines, or
2. The sum of memory demands by the VMs exceeds the total available memory on this data center, or
3. The sum of the number of required processing cores by the VMs exceeds the total number of cores on this data center.

It can happen because the mapping of VMs onto the datacenters is do neat the data center-level irrespective of schedulability consideration within the datacenters. Accordingly, the scheduler attempts to allocate all VMs in a feasible manner and if it fails, then it tries to allocate the most possible number of VMs without violation of any constraint. Finally, it returns the number of VMs (HPCAs) that could not be scheduled in this data center. Based on the value achieved from all data centers, Eq. 10 defines a penalty function to calculate the total percentage of unscheduled VMs. The penalty function is applied by the allocator to guide the search of the problem space towards the feasible solutions.

$$\mathrm{P(X)} = \frac{\sum_{i=1}^{c} \emptyset_i(\mathbf{X})}{\mathbf{N}} \tag{10}$$

where $\boldsymbol{X}$ is an assignment of VMs to the servers, and $\emptyset_i(\boldsymbol{X})$ represents the number of unscheduled VMs by the assignment $\boldsymbol{X}$ in the *i*th data center. The server-level allocation algorithm receives a mapping of VMs onto the data centers from the ICA as an input and it generates both $\boldsymbol{X}$ and $\emptyset_i(\boldsymbol{X})$.In this paper, HELP algorithm is suggested as the server-level allocation algorithm that will be explained in Section 4.3.

Besides the above mentioned two levels, the solution framework utilizes a migration handler. It aims to manage the workload distribution during runtime. After explanation of ICA and HELP, the migration handler will be discussed in details in the Section 4.4.

4.2. Imperialist Competitive Algorithm (ICA)

ICA is used as the allocator to find a right compromise between profit and CO_2 emission. ICA, a socio-politically inspired optimization strategy, was originally proposed from the work of Atashpaz-Gargari and Lucas [6]. It begins by an initial population similar to many other evolutionary algorithms. Population individuals called country are divided into two groups: colonies and imperialists. Imperialists are selected from the best countries (i.e. the lowest cost countries) and the remaining countries form the colonies. All the colonies of the initial population are divided among the imperialists based on their power. The power of an imperialist is inversely proportional to its cost. The imperialists with lower costs (i.e. higher powers) will achieve more colonies. The next step in the algorithm is moving colonies to their relevant imperialists.

The movement is a simple assimilation policy which is modeled by a directed vector from a colony to the corresponding imperialist. If the assimilation causes any colony to have a lower cost compared to its imperialist then, they will change their positions. Subsequently, the revolution process begins between the empires. Each imperialist along with its colonies form an empire. The total cost of an empire is determined by the cost of its imperialist along with the cost of its colonies. This fact is modeled by the following equation.

$$\mathrm{TC_n} = \mathrm{Cost(imperialist_n)} + \varepsilon.\,\mathrm{mean\{Cost(coloniesofempire_n)\}} \tag{11}$$

where TC_n is the total cost of the *n*th empire and ε is the colonies impact rate which is a positive number between zero and one. Increasing ε will increase the role of the colonies in determining the total power of an empire. It should be mentioned that each country (either empire or colony) is corresponding to a mapping like Figure 1 Furthermore, *C(i)* represents the cost of the *i*th mapping. To compute *C(i)*, the *i*th mapping is given as an input to the HELP algorithm and then it returns an assignment of VMs to the servers namely, X_i. Subsequently, *C(i)* can be achieved by Eq. 12.

$$\mathrm{C(i)} = \theta \mathrm{TCO_2E(X_i)} - \rho \mathrm{TProf(X_i)} + \omega \mathrm{P(X_i)} \tag{12}$$

where ω is the penalty coefficient and it is applied to scale the penalty value to the proper range. For evaluations, its value is set to 10.θ and $\rho$are the coefficients which can tune the importance of the CO_2 emission and profit respectively. All the coefficients should be selected in such a way that solving the above-mentioned problem to be equal to find a right compromise between the objectives (profit and carbon emission) while all the constraints are met.

The competition among imperialists forms the basis of the algorithm. During the competition, weak empires collapse and the most powerful ones remain. This process continues until the stopping condition is met. In the imperialistic competition, the weakest colony of the weakest empire will be exchanged from its current empire to another empire with the most likelihood to possess it. The imperialist competition will gradually result in an increase in the power of the powerful empires and a decrease in the power of the weak ones. Any empire that cannot succeed in the competition to increase its power will ultimately collapse.

The final step in the algorithm is global war. If the best imperialist in the imperialistic competition did not get any better after a certain number of iterations, the global condition is satisfied.

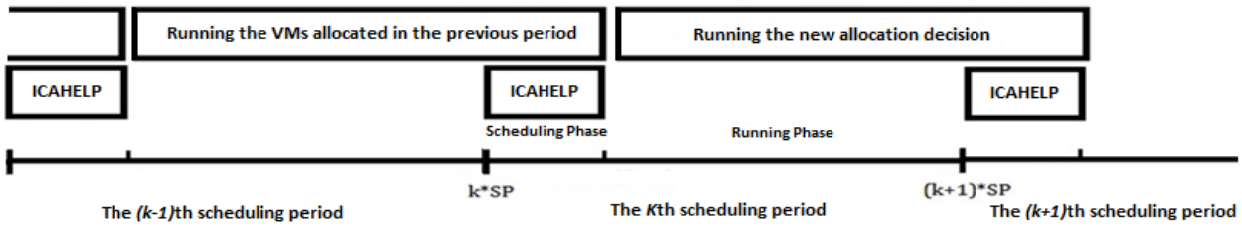

Figure 2. The *k*th scheduling period

In this way, a new empire will be formed with the same random amount of the initial population as in the initialization step. Then the best empires from the new existing empires will be selected and the algorithm repeats again. Global war can efficiently lead to escape from local optima. The algorithm stops when the stopping condition is satisfied. The stopping condition can be simply defined as the time when only one empire is left. The pseudo code of the ICA is provided in Alg. I.

ALGORITHM I. ICA

1. Initialize the empires randomly;
2. Move the colonies towards their empires (Assimilation);
3. Randomly change characteristics of some countries (Revolution);
4. **if** there is a colony which $TC_{col}<TC_{imp}$ **then**
5. Exchange the positions of that imperialist and colony;
6. **end if**
7. Compute the total cost of all empires (TCemp);
8. Pick the weakest colony from the weakest empire and give to the empire that has the most likelihood to possess it;
9. **if** there is an empire with no colonies then
10. Eliminate this empire;
11. **end if**
12. **if** there is only one empire then
13. Stop condition satisfied;
14. **else**
15. **go to** 2;
16. **end if**

In this scheduling architecture, the second-level scheduler plays a supplementary role for the high-level scheduler. As we consider an online allocation scenario, the information must be prepared quickly that for this purpose a simple heuristic is presented to find an efficient scheduling at each data center which will be described in section 4.3.

4.3. Highest Execution Time-Lowest Power Consumption (HELP) Heuristic

The execution time of HELP must be short enough to make it practically beneficial. The importance of the execution time of HELP is that it is invoked by the allocator several times. Hence, HELP is implemented based on a simple and quick idea. It schedules longer VMs (HPCAs) on the servers which have lower power consumption. As a result, the system will save energy consumption. It is worth noting that all the mentioned constraints are also taken into account by HELP.

First of all, HELP sorts the VMs according to their deadlines in ascending manner. Then, a VM is picked up from the sorted list and is scheduled on a server which has the minimum value of "HELP_Score". This metric is calculated for each VM and has a different value for each server. The various values are because of different execution time of VMs on each type of servers. The HELP_Score is computer by

$$\mathrm{HELP\_Score}_{a_i p_j} = \left(\alpha_{p_j} f_{p_j}^3 + \beta_{p_j}\right) \times e_{a_i p_j} \quad (13)$$

where $e_{a_i p_j}$ is the execution time of HPCA$a_i$and $\alpha_{p_j} f_{p_j}^3 + \beta_{p_j}$is the power consumption of thep_j.HELP attempts to schedule a VM on a server with the minimum value of the HELP_Score which is able to satisfy all the constraints. If no one of the servers can meet all the constraints, then HELP gives up the VM and tries to allocate the next VM. After assigning all the feasible VMs, the algorithm strives to allocate the remaining VMs which have been not allocated in the previous phase. For this purpose, the remaining VMs can be divided into two groups.

1. The first group contains the VMs which have been not allocated due to memory or processor limitations. The VMs in this group might be able to satisfy all the constraints if they are allocated to more than one single server. In this way, the VM covers its needs from those servers. However, in our framework to avoid a long execution time we only consider the possibility of assigning a VM to utmost two servers. In other words, if there is not available resources on two servers then we do not examine three or more servers. In this case,$HELP\_Score_{a_i p_{kj}}$ is applied to choose the proper pair of servers to run the HPCA a_i.

$$\mathrm{HELP\_Score}_{a_i p_{kj}} = \mathrm{HELP\_Score}_{a_i p_k} + \mathrm{HELP\_Score}_{a_i p_j} \quad (14)$$

Among the all feasible pair of servers, the VM will be allocated to the pair of servers that have the minimum $HELP\_Score_{a_i p_{kj}}$. For example, if the number of cores required to execute a VM is more than all the available servers within a datacenter, the VM will be allocated to the couple of servers that have enough available cores to run the VM. Therefore, it is common for an application to be scheduled on more than one server. It should be noted that if the pair of servers to which the VM is assigned have different processing powers, then the longest completion time of the VM on the assigned servers should be less than the corresponding deadline.

2. The second group contains the VMs which have been not allocated due to deadline constraint. In this group even though we tries to allocate the VM to more than one

server, it will not be able to meet its deadline because as we mentioned above, the completion time of a VM assigned to more than one server is the longest completion time of that VM.

Eventually, HELP returns the number of VMs which are not able to run on the servers of the data center.

4.4. Online Allocation

The allocation algorithm comprising both ICA and Help (let's call it ICAHELP) is frequently invoked at the start of every scheduling period to allocate the VMs waited in the execution queue. Indeed, each HPCA after to which a VM is assigned, should be inserted in the execution queue then it waits for the scheduler. The execution time of a VM could be longer than a scheduling period that in this case, unfinished VMs remaining from previous periods might be migrated from the current server to another based on the scheduler decision. The migrated VMs can continue their executions on the new servers. As a result, the algorithm allocates new incoming VMs in each period besides a potential reallocation of older VMs that have not finished their executions yet. It should be mentioned that the migration overhead is ignored in this research similar to [14].

Because, it can be imagined infinitesimal for live migration supported by pre-copying VMs onto the destination server before starting its execution.

Each scheduling period is divided into two phases, *scheduling* and *running*. Figure 2 illustrates the *k*th scheduling period in the system. As is shown in the figure, in the scheduling phase ICAHELP is executed while at the same time VMs allocated in the previous period continue to run concurrently. Just at the end of the scheduling phase, new allocation decision is made and after that the VMs are runing according to the new allocation plan. They continue to run until the end of scheduling phase in the next period. The cycle continues until all the VMs finish their execution and no new VM arrives.

To expedite the ICAHELP and to make it more applicable as an online allocation algorithm in real-world systems, a new stopping condition is considered. The algorithm is terminated once its execution time becomes more than ϑ% of the scheduling period. Therefore, the overhead of scheduling is strongly dependent on both the length of scheduling period and the value of ϑ. Therefore, there are two options to reduce the allocation overhead.

The first one is to decrease the execution time of the allocation algorithm by reduction of ϑ. The second one is to increase the scheduling period. Limiting the execution time of ICAHELP may lead to a decrease in the solution quality or it may even not be able to satisfy all the constraints for a heavy workload. Ironically, the second option is more reasonable to deal with this problem. However, long scheduling period results in some challenges related to both handling of deadlines and energy efficiency, that for both issues a separate solution is introduced in this paper.

Let's first have a look at the deadline handling solution for long scheduling periods. For this purpose, we categorize the incoming HPCAs into two main groups called *normal* and *strict*. New incoming HPCAs are examined in terms of meeting deadlines before we put them in the execution queue. If the execution time of a HPCA on the fastest processor, plus the remaining time until the next scheduling invocation, is greater than the HPCA deadline, then its corresponding VM is marked as a strict VM, and otherwise it is considered as a normal VM. If a strict VM arrives in the scheduling phase, it will be dropped. Because, during the scheduling phase we expect that the available resources in the system keep constant otherwise, ICAHELP may find an acceptable allocation which is not feasible anymore due to some changing in available resources.

On the other side, there is not enough time to delay the execution of the strict VM until the next scheduling period. Hence we have to drop them. It should be noted that this problem can be easily resolved by reserving a small part of resources to run the strict VMs arriving during scheduling phase however, it is not considered in our implemented framework.

During the running phase if a strict VM arrives, then the allocation algorithm allocates it to the best feasible server as soon as it arrives. To find the best feasible server the cost of execution of the arrival VM on all servers are calculated according to Eq. 11, and then it is allocated to the server which has the minimum value for the cost.

It is easy to imagine that whenever a normal VM arrives it is inserted in the execution queue and it waits for the next period. Although the proposed online allocation framework is able to cope with the deadlines, more maturity is still required to reach a maximum efficiency specifically for long scheduling periods. A solution to achieve more efficiency is explained in subsection 4.5.

4.5. Migration Handler

Some VMs may finish their execution before the end of the scheduling period that it can result in a significant reduction of the utilization on that server. In such situations, if the utilization of a server becomes less than a threshold it would be efficient to migrate the remaining VMs running on this server to another feasible server. Afterwards, the server can be turned off to decrease energy consumption.

The destination server can be chosen based on the best fit decision (similar to the allocation of the strict VMs) if and only if the utilization of the destination server is more than the threshold. Aprecondition is also taken into account to perform a migration operation. The precondition is that if the remaining time until the end of the scheduling period is less than the constant ε, then migrate on is not allowed to perform. The precondition is applied to avoid the migration overhead if the end of scheduling period is too close.

Some other considerations can be applied in empirical systems to adapt the proposed allocation framework with real limitations such as:

1. We can confine the migration operation only within a data center. It can be applied if the cost of migration among the data centers is not negligible.
2. A more precondition can be applied which allows migration if there is not any strict VM on the source server. In other words, migration of a strict VM is not allowed due to deadline concerns.

However, since the migration overhead is ignored in the current work, these conditions have not taken into considerations.

5. Performance Evaluation

As the target system in this research is a set of geographically distributed cloud data centers with a fairly large size, it is difficult to conduct similar and repeatable experiments on a real SGDC. To cope with this problem simulation is used as a common method in the literature. To assess the proposed framework the target system is simulated precisely considering all entities and constraints. The simulation framework is implemented on C++ while the system specification to run the simulations is: Fedora version 14 as operating system, 2.13 GHz Core i3 Intel Processor and 4 Gigabyte Dual Channel RAM.

To model the HPCAs, we use workload traces from Feitelson's Parallel Workload Archive (PWA) [14]. The PWA provides workload traces that reflect the characteristics of real parallel applications. We obtain the submit time, requested number of CPUs and actual runtime of applications from the PWA. Also, the methodology proposed by [15] is used to synthetically assign deadlines through two classes namely Low Urgency (LU) and High Urgency (HU). We suppose that 20% of all applications belong to the HU class and the other 80% belong to the LU class.

We suppose that arrival rate of services follows Poisson process and its parameter is corresponding to the service arrival rate per scheduling period. Three classes of application arrival rates are considered in our experiments, Low, Medium and High. We vary the original workload by changing the submission time of the applications. Each move from one arrival rate class to another class means that in average ten times more applications are arriving during the same period of time. Furthermore, the initial values of ICA are presented in Tab. 1.

Table 1. Initial values of ICA

Parameter	Description	Value
$N_{country}$	Number of initial countries	80
N_{imp}	Number of initial imperialists	8
R	Revolution Rate	0.1
Af	Assimilation Coefficient	2
ε	Colonies impact rate	0.02

A SGDC which is composed of eight geographically distributed data centers with different configurations is modeled as listed in Tab. 3 similar to [10, 11]. Carbon emission rates and electricity prices are derived from the data provided by US Department of Energy [16] and Energy Information Administration [17]. These values are average over the entire region that the cloud data center is located. Each data center consists of several heterogeneous servers. We consider three different types of servers which are tabulated in Tab. 2. The power related parameters are derived from a recent work presented by [10]. For the lowest frequency f_i^{min}, we use the same value as used by Garg et al. [10], i.e. the minimum frequency is 37.5% of f_i^{max}.

To evaluate the proposed algorithm, scheduling of 4026 HPCAs is simulated. The proposed solution is compared with the Genetic Algorithm which has been widely applied in the literature [11] [24]. The two algorithms are compared in three different situations. Each situation is defined by values of θ and ρ which are used to weight the objectives. Also, experiments conducted for each situation are in three different classes of service arrival rates. In the first situation ($\theta = 1, \rho = 0$), the CO_2 emission is only considered and the algorithm has attempted to minimize its value. The second situation establishes equilibrium of both objectives. Finally, the last situation considers provider's profit. The CO_2 emission is neglected in this situation and the profit is maximized.

Table 2. Characteristics of the servers

Type	CPU power factors		CPU frequency level		Disk (GB)	Memory (GB)
	β	α	f_i^{max}	f_i^{opt}		
1	65	7.5	1.8	1.630324	500	4
2	90	4.5	3.0	2.154435	600	8
3	105	6.5	3.0	2.00639	900	16

Another important factor that plays a significant role in the scenario is the considered value for the scheduling period. Considering a large value for this parameter makes the problem more complicated in the sense that the number of arrival HPCAs in a scheduling period becomes too large. On the other hand, a small value for this parameter results in a higher scheduling overhead. In order to find an efficient value for this parameter, several experiments are carried out which are represented by Figure 3. It is worth noting that the experiments reported in the Figure 3 are achieved when ϑ is set to 10%, both of θ and ρ are set to 0.5, and the arrival rate of HPCAs is selected according to the high rate shown in the Tab. 4. This figure clearly indicates that when the scheduling period is set to 80 then the best result in terms of the cost value mentioned by Eq. 11 is acquired.

Table 3. Characteristics of the cloud data centers

Location	CO_2 Emission rate (kg/kWh)	Electricity Price ($/kWh)	COP	# of Servers
New York, USA	0.389	0.15	3.052	2050
Pennsylvania, USA	0.574	0.09	1.691	2600
California, USA	0.275	0.13	2.196	650
Ohio, USA	0.817	0.09	1.270	540
North Carolina, USA	0.563	0.07	1.843	600
Texas, USA	0.664	0.1	1.608	350
France	0.083	0.17	0.915	200
Australia	0.924	0.11	3.099	250

Let's investigate the scheduling period in more details. Since the ICAHELP execution time is just ϑ% of the scheduling period, once the scheduling period is a short value like 20 or 40 as we already anticipated, ICAHELP does not have enough time to find a good solution in the problem space. By growth of the scheduling period although the quality of solutions obtained by ICAHELP is improved, the cost function goes towards unacceptable cost values. There are two main reasons for the cost increase.

Table 4. Simulation results

	HPCA arrival rate	Proposed algorithm			Genetic Algorithm		
		Avg. Profit ($)	**Avg. CO_2** (kg)	**Avg. Execution Time** (ms)	**Avg. Profit** ($)	**Avg. CO_2** (kg)	**Avg. Execution Time** (ms)
θ = 1 ρ = 0	Low	2406443	9571	42	2165798	10528	46
	Medium	8502658	36349	58	6802126	43618	65
	High	23541377	148057	203	22364308	177668	231
θ = 0.5 ρ = 0.5	Low	3526578	12083	43	3350249	13291	48
	Medium	10628601	58117	62	9034310	69742	69
	High	28681040	283250	199	25812936	368225	221
θ = 0 ρ = 1	Low	4715584	18088	41	3772467	19896	47
	Medium	12920499	77207	65	11628449	84927	71
	High	32523725	382502	213	30897539	459002	235

The first one is that for large scheduling periods the most of incoming HPCAs are labeled as a strict VM and thus they are scheduled by the best fit algorithm while we know that the best fit does not consider all possible combinations and it is not an optimal algorithm. The second reason is that as the number of incoming HPCAs in a longer scheduling period increases the problem space becomes bigger and more complex.

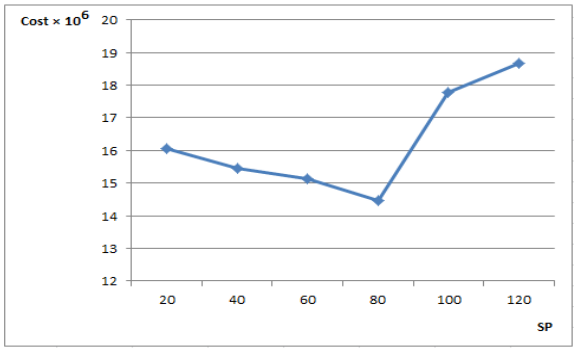

Figure 3. The cost value achieved by the proposed framework for various scheduling periods

For each situation, both algorithms (ICAHELP and GA) are run until all the applications in the given workload are scheduled. The number of scheduling periods required to schedule all applications in the workload depends to three factors, the number of applications in the workload, the submission time of applications, and the length of the scheduling period. In the simulations, for each situation, both two algorithms run 20 times to reach 95% confidence interval, and finally the average of achieved profit and carbon emission are reported in Tab. 4. The results indicate that the proposed algorithm generates better solutions in comparison with the traditional GA approach. The proposed scheduling architecture outperforms 9% in terms of the profit. Additionally, its average execution time is shorter than GA for all experiments. It should be noted that the increase in execution time of each situation is related to the increase in the number of arrival services which makes the problem space larger. In addition, the generated results are in form of Pareto solutions. Consequently, the system designers can choose appropriate θ and ρ values to reach an acceptable level of CO_2 emission and profit. In the other words, cloud providers will be able to present flexible infrastructures with the lowest cost and destructive environmental impacts.

6. Conclusion and Future Works

In this paper, the problem of scheduling HPC applications on a set of heterogeneous data centers which are located all over the world is investigated. The problem has two objectives, minimizing CO_2 emissions and maximizing cloud provider's profit. We used energy efficiency metrics of each data center such as CO_2 emission rate and COP, which change from one location to another. As the solution, a two-level scheduling algorithm is proposed which combines two meta-heuristic and heuristic algorithms.

The first level scheduler, federation-level, utilizes ICA to solve its bi-objective optimization problem. Due to heterogeneity of the cloud data centers, the scheduling decision making is directly related to the servers which the applications are scheduled on. Therefore, the second level scheduler, data center-level, schedules its assigned applications and provides required information for the federation-level scheduler.

The proposed approach is simulated precisely and has been evaluated using realistic workload traces. Based on the results, the proposed scheduling approach outperforms other mentioned related work which is based on Genetic Algorithm. For future work, we plan to progress in two different directions, the first one is to integrate Dynamic Voltage Scaling (DVS) techniques to save more energy. The second direction is to investigate the effect of the energy consumption of network equipment on both profit and carbon emission that in this case the cost of live migration cannot be easily disregarded and must be considered as overhead on the network energy consumption.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," *Journal of Future Generation Computer Systems,* vol. 25, no. 6, pp. 599-616, 2009.

[2] W. Voorsluys, J. Broberg, and R. Buyya, *Cloud Computing: Principles and Paradigms*: John Wiley and Sons, 2010.

[3] C. Belady, "In the Data Center, Power and Cooling Costs More Than the Equipment It Supports," http://www.electronics-cooling.com/2007/02/in-the-data, center-power-and-cooling-costs-more-than-the-it-equipment-it-supports, September 2014.

[4] R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-Efficient Management of Data Center Resources for Cloud Computing: a Vision, Architectural Elements, and Open Challenges," *Proc. IEEE Intl Conf. Parallel and Distributed Processing Techniques and Applications,* pp. 12-15, 2010.

[5] C. Petty, "Gartner Estimates ICT Industry Accounts for 2 Percent of Global CO2 Emission," http://www.gartner.com/it/page.jsp?id=503867, June 2007.

[6] E. Atashpaz-Gargari, and C. Lucas, "Imperialist Competitive Algorithm: An algorithm for Optimization Inspired by Imperialistic Competition," *Proc. IEEE Intl Cong. Evolutionary Computation,* pp. 4661-4667, 2007.

[7] A. Orgerie, L. Lefèvre, and J. Gelas, "Save Watts in Your Grid: Green Strategies for Energy-aware Framework in Large Scale Distributed Systems," *Proc. IEEE Intl Conf. Parallel and Distributed Systems*, pp. 171-178, 2008.

[8] C. Patel, R. Sharma, C. Bash, and S. Graupner, *Energy aware Grid: Global Workload Placement Based on Energy Efficiency,* Technical Report, Palo Alto: HP Labs, 2002.

[9] S. Garg, C. Yeo, and R. Buyya, "Green Cloud Framework for Improving Carbon Efficiency of Clouds," *Proc. IEEE Intl Conf. Parallel Processing,* pp. 491-502, 2011.

[10] S. Garg, C. Yeo, A. Anandasivam, and R. Buyya, "Environment-conscious Scheduling of HPC Applications on Distributed Cloud-oriented Data Centers," *Journal of Parallel and Distributed Computing,* vol. 71, no. 6, pp. 732-749, 2011.

[11] Y. Kessaci, N. Melab, and E. Talbi, "A Pareto-based Metaheuristic for Scheduling HPC Applications on a Geographically Distributed Cloud Federation," *Proc. IEEE Intl Conf. Cluster Computing*, pp. 1-21, 2012.

[12] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, "Making Scheduling Cool: Temperature-aware Workload Placement in Data Centers," *Proc. IEEE Intl Conf.* USENIX, pp. 61-75, 2005.

[13] Q. Tang, S. K. S. Gupta, D. Stanzione, and P. Cayton, "Thermal-aware Task Scheduling to Minimize Energy Usage of Blade Server Based Data Centers," *Proc. IEEE Intl Symp. Dependable, Autonomic and Secure Computing,* pp. 195-202, 2006.

[14] D. Feitelson, "Parallel Workloads Archive," http://www.cs.huji.ac.il/labs/parallel/workload, August 2009.

[15] D. Irwin, L. Grit, and J. Chase, "Balancing Risk and Reward in a Market-based Task Service," *Proc. IEEE Intl Symp. High Performance Distributed Computing,* pp. 160-169, 2004.

[16] US Department of Energy, "Voluntary Reporting of Greenhouse Gases: Appendix F Electricity Emission Factors," http://agroclimate.org/tools/tabs_carbonfootprint/references, September 2007.

[17] US Department of Energy, "US Energy Information Administration (EIA) Report," http://www.eia.doe.gov/cneaf/electricity/epm/table5_6_a.html, September 2007.

[18] H. R. Faragardi, A. Rajabi, and R. Shojaee, "Towards Energy-Aware Resource Scheduling to Maximize Reliability in Cloud Computing Systems with QoS Guarantee," *Proc. IEEE Intl Conf. High Performance Computing and Communications,* pp. 61-67, 2013.

[19] R. Rajabi, H. Faragardiand, and T. Nolte "An Efficient Scheduling of HPC Applications on Geographically Distributed Cloud Data Centers," *Proc. IEEE Intl Symp. Computer Networks and Distributed Systems,* pp. 50-56, 2013.

[20] A. Khosravi, S. K. Garg, and R. Buyya, "Energy and Carbon-efficient Placement of Virtual Machines in Distributed Cloud Data Centers," *Proc. IEEE Intl Conf. Parallel Processing,* pp. 317-328, 2013.

[21] B. Wadhwa, and V. Amandeep, "Energy and Carbon Efficient VM Placement and Migration Technique for Green Cloud Data Centers," *Proc. IEEE Intl Conf. Contemporary Computing*, pp. 826-831, 2014.

[22] C. Jing, Z. Yanmin, and L. Minglu, *Customer Satisfaction-aware Scheduling for Utility Maximization on Geo-distributed Data Centers,* Concurrency and Computation: Practice and Experience, 2014.

[23] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing," *Journal of Future Generation Computer Systems,* vol. 28, no. 5, pp. 755-768, 2012.

[24] M. Mohand, N. Melab, Y. Kessaci, Y. C. Lee, E-G. Talbi, A. Y. Zomaya, and D. Tuyttens, "A Parallel Bi-objective Hybrid Meta-heuristic for Energy-aware Scheduling for Cloud Computing Systems," *Journal of Parallel and Distributed Computing,* vol. 71, no. 11, pp. 1497-1508, 2011.

Hamid Reza Faragardi received his M.Sc. from University of Tehran in 2012 in Computer Engineering. He currently is a PhD candidate at the real-time department of Malardalen University in Sweden. His main research interests comprise cloud computing, real-time scheduling and optimization problems. His Ph.D. project is development of AUTOSAR for multi core platforms in automotive systems

where Volvo and ABB are involved in the project as the industrial partners. He has already published several conference papers and journal articles which most of them concentrate on energy aware scheduling in Cloud computing systems.

E-mail: hamid.faragardi@mdh.se

Aboozar Rajabi received his M.Sc. from University of Tehran in 2013 in Computer Engineering. His main research interests include cloud computing, green computing and game theory. He has published several conference papers which most of them cover green computing in the context of Cloud systems.

E-mail: ab.rajabi@ut.ac.ir

Thomas Nolte Thomas Nolte was awarded a B.Eng., M.Sc., Licentiate, and Ph.D. degree in Computer Engineering from Malardalen University (MDH), Vasteras, Sweden, in 2001, 2002, 2003, and 2006, respectively. He has been a Visiting Researcher at University of California, Irvine (UCI), Los Angeles, USA, in 2002, and Visiting Researcher at University of Catania, Italy, in 2005. He has been a Postdoctoral Researcher at University of Catania in 2006, and at MDH in 2006-2007. Thomas Nolte became Full Professor of Computer Science in 2012. He has published more than 150 papers in peer-reviewed journals and conferences. His research interests include predictable execution of embedded systems, design, modeling and analysis of real-time systems, multi core systems, distributed embedded real-time systems.

E-mail: thomas.nolte@mdh.se

Amir Hossein Heidarzadeh is a M.Sc. student in financial engineering at Malardalen University at Sweden. He received a two bachelor degree in both Applied Mathematics and Economics. His main research interests include game theory and optimization.

E-mail: amir.heidarizadeh@mdh.se

Paper Handling Data:

Submitted: 25.05.2014
Received in revised form: 16.09.2014
Accepted: 07.10.2014
Corresponding author: Hamid Reza Faragardi,
School of Innovative Design and Engineering, Malardalen University, Sweden.