# An Efficient Scheduling of HPC Applications on Geographically Distributed Cloud Data Centers

Aboozar Rajabi[1], Hamid Reza Faragardi[2], Thomas Nolte[2]

[1]School of Electrical and Computer Engineering, University of Tehran, Iran
[2]Mälardalen Real-Time Research Centre, Mälardalen University, Sweden
ab.rajabi@ut.ac.ir, {hamid.faragardi, thomas.nolte}@mdh.se

**Abstract.** Cloud computing provides a flexible infrastructure for IT industries to run their High Performance Computing (HPC) applications. Cloud providers deliver such computing infrastructures through a set of data centers called a cloud federation. The data centers of a cloud federation are usually distributed over the world. The profit of cloud providers strongly depends on the cost of energy consumption. As the data centers are located in various corners of the world, the cost of energy consumption and the amount of $CO_2$ emission in different data centers varies significantly. Therefore, a proper allocation of HPC applications in such systems can result in a decrease of $CO_2$ emission and a substantial increase of the providers' profit. Reduction of $CO_2$ emission also mitigates the destructive environmental impacts. In this paper, the problem of scheduling HPC applications on a geographically distributed cloud federation is scrutinized. To address the problem, we propose a two-level scheduler which is able to reach a good compromise between $CO_2$ emission and the profit of cloud provider. The scheduler should also satisfy all HPC applications' deadline and memory constraints. Simulation results based on a real intensive workload indicate that the proposed scheduler reduces the $CO_2$ emission by 11% while at the same time it improves the provider's profit in average.

**Keywords:** Cloud Computing, Data Center, Energy-aware scheduling, $CO_2$ emission, Multi-objective optimization

## 1 Introduction

Cloud computing has emerged as a new paradigm to support the utility computing idea [1]. The advantages of this computing model motivate the IT industries, which require a High Performance Computing (HPC) infrastructure to run IT applications, to utilize this model and have access to a flexible HPC infrastructure. 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 affects the cost of services. Based on a set of recent estimations, energy consumption strongly

contributes to the total operational cost of data centers [3]. Higher energy consumption would make the services more expensive. 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].

In this paper, the problem of scheduling HPC applications in the geographically distributed data centers (i.e., a cloud federation) is investigated. The goal is to schedule a set of HPC applications in such a way that carbon emission is minimized and the profit of the cloud provider is maximized. As the locations of data centers are spread over the world, the energy efficiency factors such as electricity cost, carbon emission rate and Coefficient of Performance (COP) usually vary across various data centers. In addition, in order to provide a solution suitable for real-time applications, the proposed scheduler attempts to meet application deadlines. Moreover, memory constraint is considered to present a realistic solution that can run on cloud federation hardware. The problem is formulated as an Integer Linear Programming (ILP) and then in order to cope with the ILP problem, a two-level scheduling algorithm is introduced. The scheduler is an online scheduling approach in which the federation-level is the first level scheduler and The Highest Execution time-Lowest Power consumption (HELP) scheduler is the second one. The former scheduler is 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. The latter scheduler is a greedy heuristic algorithm employed as a local scheduler at each data center. These two schedulers cooperate with each other in such a way that the memory and deadline constraints are satisfied while achieves a right compromise between the profit and the amount of $CO_2$ emission.

The main contributions of this paper can be expressed as: 1) Proposing an online scheduling approach which works based on a two-level scheduling architecture, 2) Considering heterogeneous resources, 3) Contemplating real-time HPC applications and various system constraints.

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. $CO_2$ 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]. They have also proposed some heuristics for scheduling of HPC workloads in several data centers [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 using a meta-heuristic approach. They have proposed a meta-scheduler using a multi-objective genetic algorithm to optimize the objectives. Both mentioned studies, consider a homogeneous data center in which the servers are the same despite the difference with other data centers' servers.

This work is different from the mentioned works, because it addresses the problem of scheduling HPC workloads in heterogeneous cloud data centers. The proposed method uses a two level scheduler to make the scheduling decisions. The scheduler considers both the $CO_2$ emission of each data center and the profit of cloud provider.

# 3 Problem Definition

## 3.1 System Model

The system is a set *C* of *c* data centers which compose a cloud federation. 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 data centers directly impact on the complexity of the problem. Each data center is a set *P* of *p* heterogeneous servers. Each server also has a specific amount of memory.

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. A set *A* consists of *N* elements represents the applications. Each application has a deadline that must be met. A user submits his requirement for an application $a_i$ 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 deadline to complete $a_i$, $n_{a_i}$ is the number of CPUs needed to execute $a_i$, $e_{a_i p_j}$ is a vector 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 $a_i$.

## 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 CUP 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

$$P = A'C'V^2f + I_{leak}V + P_{short} \tag{1}$$

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 $\alpha$. 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, $V^2f$ can be replaced by $f^3$. Therefore, the energy consumption of the computing equipment for execution of $a_i$ is computed by

$$E^{CE}_{a_ic_k}(X) = \sum_{j=0}^{p}(\alpha_{p_j}f_{p_j}^3 + \beta_{p_j}) \times e_{a_ip_j}\, x_{ij} \tag{2}$$

where $x_{ij}$ 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 service 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 our scheduling period. The energy consumption of the cooling equipment of the data center $c_j$, $E^{CS}_{c_j}$, is defined by

$$E^{CS}_{c_j} = E^{P}_{c_j} / COP_{c_j} \tag{3}$$

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

$$E^{total}_{a_ic_j} = E^{CS}_{a_ic_j} + E^{CE}_{a_ic_j} = (1 + 1/COP_{c_j}) \times E^{CE}_{a_ic_j} \tag{4}$$

### 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 each cloud data center. The $CO_2$ emission due to the execution of application $a_i$ on the data center $c_j$ is computed by

$$CO_2E_{a_ic_j} = r_{c_j}^{CO_2} \times E_{a_ic_j}^{total} \tag{5}$$

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

$$TCO_2E\ (X) = \sum_{i=0}^{N} \sum_{j=0}^{c} CO_2E_{a_ic_j}(X) \tag{6}$$

### 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

$$Prof_{a_ic_j} = n_{a_i} \times e_{a_ic_j} \times p_{a_i}^{c} - p_{c_j}^{e} \times E_{a_ic_j}^{total} \tag{7}$$

where $e_{a_ic_j}$ 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 $E_{a_ic_j}^{total}$ is the total energy consumption of $a_i$ on $c_j$. Therefore, the total profit can be computed as $TProf(X)$ by

$$TProf(X) = \sum_{i=0}^{a} \sum_{j=0}^{c} Prof_{a_ic_j}(X) \tag{8}$$

## 4 Proposed Solution

In this section, an online scheduling algorithm is suggested to cope with the mentioned problem. The architecture of the proposed solution consists of a two-level scheduling algorithm. In the following, the architecture is explained in details and it is demonstrated how this scheduling architecture can be applied.

### 4.1 Architecture

**Federation-level Scheduler**: This scheduler is located at the high level to partition a set of applications among the available data centers in a cloud federation. In this

level, the applications 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 federation level, the high-level scheduler should be aware of servers which are executing the applications. However, in most of the previous studies only homogenous data centers are taken into account in which the high-level scheduler does not need to be aware of data center-level scheduling. Because, if the servers of a data center are the same, calculating the energy consumed by the application is not dependent to which server of this data center is executing the application. Nevertheless, we introduce an intelligent architecture which is able to separate these two scheduling levels even for heterogeneous data centers and provides us a two-level scheduler. The federation-level scheduler 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 services 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*. Fig. 2 shows an illustrative example for a mapping of services. The third element of this example is two, which means that the third application is mapped to the second data center.

| $a_1$ | $a_2$ | $a_3$ | | $a_N$ |
|---|---|---|---|---|
| 1 | 1 | 2 | … | 1 |

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

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

**Data Center-level Scheduler**: Each data center has a local scheduler. The submitted applications to this data center are scheduled by the corresponding scheduler. The scheduler at this level aims to find an allocation which can meet the memory and deadline constraints while at same time it attempts to minimize the energy consumption. Decreasing the energy consumption potentially leads to mitigation of $CO_2$ emission and increasing the cloud provider profit. It should be noted that the data center-level scheduler may not be able to find a feasible allocation. In other words, if all the applications mapped to this data center are allocated to its servers, then some applications may miss their deadlines or the sum of memory demands by the applications exceeds the available memory on the servers. It can happen because the mapping of services onto the data centers is done at the federation-level irrespective of schedulability consideration within the data centers. Accordingly, the second-level scheduler attempts to allocate all services in a feasible manner and if it fails, then it tries to allocate the most possible number of services without violation of the constraints. Finally, it returns the number of services that could not be scheduled in this data center. Based on the value achieved from all data centers, Eq. 9 defines a penalty function to calculate the total percentage of unscheduled services.

$$P(X) = \frac{\sum_{i=1}^{c} \emptyset_i(X)}{N} \tag{9}$$

where $X$ is an assignment of services to the servers, and $\emptyset_i(X)$ represents the number of unscheduled services by the assignment $X$ in the *i*th data center. The second-level

scheduler receives a mapping of services onto the data centers from the ICA as an input and it generates both $X$ and $\emptyset_i(X)$. The HELP algorithm is suggested in this paper as the data center scheduler and it will be explained in Section 4.3.

### 4.2 Imperialist Competitive Algorithm (ICA)

ICA is used 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.

$$TC_n = Cost(imperialist_n) + \varepsilon.mean\{Cost(colonies\ of\ empire_n)\} \quad (10)$$

where $TC_n$ is the total cost of the *n*th empire and $\varepsilon$ is the colonies impact rate which is a positive number between zero and one. Increasing $\varepsilon$ 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 Fig. 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 services to the servers namely, $X_i$. Subsequently, *C(i)* can be achieved by Eq. 11.

$$C(i) = \theta\, TCO_2E\,(X_i) - \rho\, TProf(X_i) + \omega\, P(X_i) \quad (11)$$

where $\omega$ 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. $\theta$ 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 iteration time, the global condition is satisfied. 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. It can be simply defined as the time when only one empire is left. The pseudo code of the ICA is provided in Alg. I.

In this scheduling architecture, the second-level scheduler plays a supplementary role for the high-level scheduler. As the information must be prepared quickly, a simple heuristic is proposed to solve the optimization problem at each data center.

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 em-
pire 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
```

### 4.3 Highest Execution time-Lowest Power consumption (HELP) Heuristic

The execution time of HELP should be short enough to make it practically beneficial. Indeed, its run time must be admissible because the federation-level scheduler would run it several times. Hence, HELP is implemented based on a simple and quick idea. It schedules longer applications on the servers which have lower power consumption. As a result, the system will save energy consumption. It is worth noting that the mentioned application and system constraints are also taken into account by HELP.

First of all, HELP sorts the applications according to their deadlines in ascending manner. Then, an application 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 application and has a different value for each server. The various values are because of different execution time of each application on each server's type. The "HELP Score" is computer by

$$HELP\_Score_{a_i p_j} = (\alpha_{p_j} f_{p_j}^3 + \beta_{p_j}) \times e_{a_i p_j} \quad (12)$$

where $e_{a_i p_j}$ is the execution time of application $a_i$ and $\alpha_{p_j} f_{p_j}^3 + \beta_{p_j}$ is the power consumption of the $p_j$. HELP attempts to schedule an application 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 the constraints, then HELP leaves the application and tries to allocate the next application. In addition, if the application requires more than one server, it will cover its needs from other servers. Therefore, it is common for an application to be scheduled on more than one server. In this case, in order to satisfy the application's deadline, the longest completion time of the application on the assigned servers should be less than the corresponding deadline.

## 5 Performance Evaluation

As the proposed solution is designed for cloud federations, it is essential to perform evaluations on large-scale cloud data centers. However, it is difficult to conduct similar and repeatable experiments on a real cloud federation. To cope with problem, simulation has used as a common solution to evaluate energy-aware scheduling. Thus, the addressed system is simulated precisely considering all entities and constraints.

To model the HPC applications, 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. In addition, three classes of application arrival rates are considered in our experiments, Low, Medium and High. We vary the original workload by changing the submit time of the applications. Each move from an arrival rate class to another means ten times more applications are arriving during the same period of time. In the other words, each time we divide the submission time by 10. Furthermore, the initial values of ICA are presented in Tab. 1. Finally, the scheduling period in our algorithm is set to 50s.

**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 |
| $\varepsilon$ | Colonies impact rate | 0.02 |

A cloud federation which is composed of 8 geographically distributed data centers with different configurations is modeled as listed in Tab. 2 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. 3. 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 HPC applications is simulated. The proposed solution is compared with the Genetic Algorithm proposed earlier by Y. Kessaci [11]. The two algorithms are compared in three different situations. Each situation is defined by values of θ and $\rho$ 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 cloud data centers

| **Location** | **$CO_2$ Emission rate** (kg/kWh) | **Electricity Price** ($/kWh) | **COP** | **Number 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 |

As the Tab. 4 indicates, the proposed algorithm produces 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.

**Table 3.** Table 4. Characteristics of the servers

| **Type** | **CPU power factors** | | **CPU frequency level** | | **Disk** (GB) | **Memory** (GB) |
|---|---|---|---|---|---|---|
| | $\beta$ | $\alpha$ | $f_i^{max}$ | $f_i^{opt}$ | | |
| 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 |

**Table 5.** Simulation results

| | Service 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) |
| $\theta = 1$ $\rho = 0$ | Low | 2406443 | 9571 | 42 | 2165798 | 10528 | 46 |
| | Medium | 8502658 | 36349 | 58 | 6802126 | 43618 | 65 |
| | High | 23541377 | 148057 | 203 | 22364308 | 177668 | 231 |
| $\theta = 0.5$ $\rho = 0.5$ | Low | 3526578 | 12083 | 43 | 3350249 | 13291 | 48 |
| | Medium | 10628601 | 58117 | 62 | 9034310 | 69742 | 69 |
| | High | 28681040 | 283250 | 199 | 25812936 | 368225 | 221 |
| $\theta = 0$ $\rho = 1$ | Low | 4715584 | 18088 | 41 | 3772467 | 19896 | 47 |
| | Medium | 12920499 | 77207 | 65 | 11628449 | 84927 | 71 |
| | High | 32523725 | 382502 | 213 | 30897539 | 459002 | 235 |

In addition, the generated results are in form of Pareto solutions. Consequently, the system designers can choose appropriate $\theta$ and $\rho$ 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 works, we plan to integrate Dynamic Voltage Frequency Scaling (DVFS) techniques to save more energy.

# 7 References


1. Buyya, R., S. Yeo, C., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems, vol. 25, no. 6, pp. 599–616, Jun. (2009)
2. W. Voorsluys, J. Broberg, and R. Buyya, "Introduction to Cloud Computing," Cloud Computing: Principles and Paradigms, 1-41pp, R. Buyya, J. Broberg, A.Goscinski (eds), ISBN-13: 978-0470887998, Wiley Press, New York, USA, February (2011)
3. Belady, C.: In the data center, power and cooling costs more than the IT equipment it supports. [Online]. Available: http://www.electronics-cooling.com/articles/2007/feb/a3/. [Accessed: 1-Sep-2013].
4. Buyya, R., Beloglazov, A., Abawajy, J.: Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges. In: Proceedings of the 2010 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2010), Las Vegas, USA, July 12–15, (2010)
5. Gartner, Gartner estimates ICT industry accounts for 2 percent of global $CO_2$ emissions, Apr. 2007. http://www.gartner.com/it/page.jsp?id=503867.
6. Atashpaz-Gargari, C. Lucas, E.: Imperialist Competitive Algorithm: An algorithm for optimization inspired by imperialistic competition. IEEE Congress on Evolutionary Computation, (2007)
7. Orgerie, A., Lefèvre, L., Gelas, J.: Save watts in your grid: green strategies for energy-aware framework in large scale distributed systems. In: Proceedings of the 2008 14th IEEE International Conference on Parallel and Distributed Systems, Melbourne, Australia, (2008)
8. Patel, C., Sharma, R., Bash, C., Graupner, S.: Energy aware grid: global workload placement based on energy efficiency. Technical Report HPL-2002-329, HP Labs, Palo Alto, Nov. (2002)
9. Garg, S., Yeo, C., Buyya, R.: Green cloud framework for improving carbon efficiency of clouds. In: Proc. of the 17th International Conference on Parallel Processing, pp. 491–502 (2011)
10. Garg, S., Yeo, C., Anandasivam, A., Buyya, R.: Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers. Journal of Parallel and Distributed Computing 71(6), 732–749 (2011)
11. Kessaci, Y., Melab, N., Talbi, E.: A Pareto-based metaheuristic for scheduling HPC applications on a geographically distributed cloud federation. Cluster Computing, pp. 1–21, (2012)
12. Moore, J., Chase, J., Ranganathan, P., Sharma, R.: Making scheduling ''cool'': temperature-aware workload placement in data centers. In: Proceedings of the 2005 Annual Conference on USENIX Annual Technical Conference, Anaheim, CA, (2005)
13. Tang, Q., Gupta, S.K.S., Stanzione, D., Cayton, P.: Thermal-aware task scheduling to minimize energy usage of blade server based datacenters. In: Proceedings of the 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing, DASC 2006, IEEE Computer Society, Los Alamitos, CA, USA, (2006)
14. Feitelson, D., "Parallel workloads archive", Aug. 2009. Available: http://www.cs.huji.ac.il/labs/parallel/workload. [Accessed: 7-May-2013].
15. D. Irwin, L. Grit, J. Chase, Balancing risk and reward in a market-based task service, in: Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing, Honolulu, USA, 2004.
16. US Department of Energy, Voluntary reporting of greenhouse gases: Appendix F. Electricity emission factors, 2007. http://www.eia.doe.gov/oiaf/1605/pdf/ Appendix20F_r071023.pdf.
17. US Department of Energy, US Energy Information Administration (EIA) report, 2007. http://www.eia.doe.gov/cneaf/electricity/epm/table5_6_a.html.