J. Parallel Distrib. Comput. 71 (2011) 732–749

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Environment-conscious scheduling of HPC applications on distributed Cloud-oriented data centers

Saurabh Kumar Garg [a,*], Chee Shin Yeo [b], Arun Anandasivam [c], Rajkumar Buyya [a]

[a] Cloud Computing and Distributed Systems Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, Australia

[b] Advanced Computing Programme, Institute of High Performance Computing, Singapore

[c] Institute of Information Systems and Management, Karlsruhe Institute of Technology, Germany

ARTICLE INFO

Article history:
Received 3 September 2009
Received in revised form
13 April 2010
Accepted 30 April 2010
Available online 11 May 2010

Keywords:
Cloud computing
High Performance Computing (HPC)
Energy-efficient scheduling
Dynamic Voltage Scaling (DVS)
Green IT

ABSTRACT

The use of High Performance Computing (HPC) in commercial and consumer IT applications is becoming popular. HPC users need the ability to gain rapid and scalable access to high-end computing capabilities. Cloud computing promises to deliver such a computing infrastructure using data centers so that HPC users can access applications and data from a Cloud anywhere in the world on demand and pay based on what they use. However, the growing demand drastically increases the energy consumption of data centers, which has become a critical issue. High energy consumption not only translates to high energy cost which will reduce the profit margin of Cloud providers, but also high carbon emissions which are not environmentally sustainable. Hence, there is an urgent need for energy-efficient solutions that can address the high increase in the energy consumption from the perspective of not only the Cloud provider, but also from the environment. To address this issue, we propose near-optimal scheduling policies that exploit heterogeneity across multiple data centers for a Cloud provider. We consider a number of energy efficiency factors (such as energy cost, carbon emission rate, workload, and CPU power efficiency) which change across different data centers depending on their location, architectural design, and management system. Our carbon/energy based scheduling policies are able to achieve on average up to 25% of energy savings in comparison to profit based scheduling policies leading to higher profit and less carbon emissions.

1. Introduction

During the last few years, the use of High Performance Computing (HPC) infrastructure to run business and consumer based IT applications has increased rapidly. This is evident from the recent Top500 supercomputer applications where many supercomputers are now used for industrial HPC applications, including 9.2% of them are used for Finance and 6.2% for Logistic services [58]. Thus, it is desirable for IT industries to have access to a flexible HPC infrastructure which is available on demand with minimum investment. Cloud computing [10] promises to deliver such reliable services through next-generation data centers built on virtualized compute and storage technologies. Users are able to access applications and data from a "Cloud" anywhere in the world on demand and pay based on what they use. Hence, Cloud computing is a highly scalable and cost-effective infrastructure for running HPC applications which requires ever-increasing computational resources.

However, Clouds are essentially data centers that require high energy[1] usage to maintain operation [5]. Today, a typical data center with 1000 racks need 10 MW of power to operate [50]. High energy usage is undesirable since it results in high energy cost. For a data center, the energy cost is a significant component of its operating and up-front costs [50]. Therefore, Cloud providers want to increase their profit or Return on Investment (ROI) by reducing their energy cost. Many Cloud providers are thus building different data centers and deploying them in many geographical locations so as not only to expose their Cloud services to business and consumer applications, e.g. Amazon [1], but also to reduce energy cost, e.g. Google [40].

In April 2007, Gartner estimated that the Information and Communication Technologies (ICT) industry generates about 2% of the total global CO_2[2] emissions, which is equal to the aviation industry [30]. As governments impose carbon emissions limits on the ICT

* Corresponding author.
E-mail addresses: sgarg@csse.unimelb.edu.au (S.K. Garg), yeocs@ihpc.a-star.edu.sg (C.S. Yeo), anandasivam@kit.edu (A. Anandasivam), raj@csse.unimelb.edu.au (R. Buyya).

0743-7315/$ – see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2010.04.004

[1] Energy and electricity is used interchangeably.

[2] CO_2 and carbon is used interchangeably.

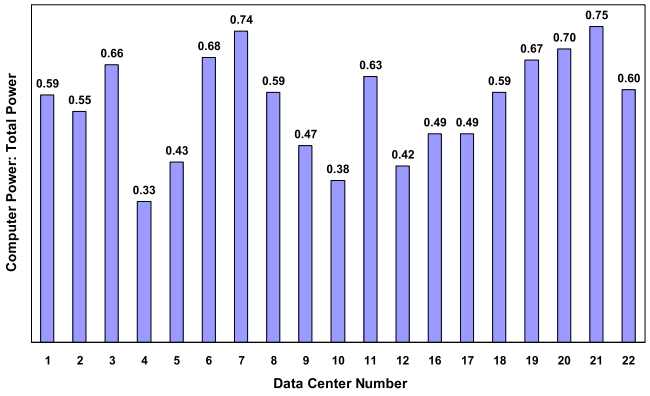


Fig. 1. Computer power consumption index.
Source: [32].

industry like in the automobile industry [18,21], Cloud providers must reduce energy usage to meet the permissible restrictions [15]. Thus, Cloud providers must ensure that data centers are utilized in a carbon-efficient manner to meet scaling demand. Otherwise, building more data centers without any carbon consideration is not viable since it is not environmentally sustainable and will ultimately violate the imposed carbon emissions limits. This will in turn affect the future widespread adoption of Cloud computing, especially for the HPC community which demands scalable infrastructure to be delivered by Cloud providers. Companies like Alpiron [2] already offer software for cost-efficient server management and promise to reduce energy cost by analyzing, via advanced algorithms, which server to shutdown or turn on during the runtime.

Motivated by this practice, this paper enhances the idea of cost-effective management by taking both the aspects of economic (profit) and environmental (carbon emissions) sustainability into account. In particular, we aim to examine how a Cloud provider can achieve optimal energy sustainability of running HPC workloads across its entire Cloud infrastructure by harnessing the heterogeneity of multiple data centers geographically distributed in different locations worldwide.

The analysis of previous work shows that little investigation has been done for both economic and environmental sustainability to achieve energy efficiency on a global scale as in Cloud computing. First, previous work has generally studied how to reduce energy usage from the perspective of reducing cost, but not how to improve the profit while reducing the carbon emissions which is also significantly impacting the Cloud providers [25]. Second, most previous work has focused on achieving energy efficiency at a *single* data center location, but not across *multiple* data center locations. However, Cloud providers such as Amazon EC2 [1] typically has multiple data centers distributed worldwide. As shown in Fig. 1, the energy efficiency of an individual data center in different locations changes dynamically at various times depending on a number of factors such as energy cost, carbon emission rate, workload, CPU power efficiency, cooling system, and environmental temperature. Thus, these different contributing factors can be considered to exploit the heterogeneity across multiple data centers for improving the overall energy efficiency of the Cloud provider. Third, previous work has mainly proposed energy saving policies that are application-specific [26,28], processor-specific [52,17], and/or server-specific [60,38]. But, these policies are only applicable or most effective for the specific models that they are specially designed for. Hence, we propose some simple, yet effective generic energy-efficient scheduling policies that can be extended to any application, processor, and server models so that they can be readily deployed in existing data centers with minimum changes. Our generic scheduling policies within a data center can also easily complement any of these application-specific, processor-specific, and/or server-specific energy saving policies that are already in place within existing data centers or servers.

Hence, the key contributions of this paper are:

(1) A novel mathematical model for energy efficiency based on various contributing factors such as energy cost, carbon emission rate, HPC workload, and CPU power efficiency;
(2) Near-optimal energy-efficient scheduling policies which not only minimize the carbon emission and maximize the profit of the Cloud provider, but also can be readily implemented without much infrastructure changes such as the relocation of existing data centers;
(3) Energy efficiency analysis of our proposed policies (in terms of carbon emissions and profit) through extensive simulations using real HPC workload traces, and data center carbon emission rates and energy costs to demonstrate the importance of considering various contributing factors;
(4) Analysis of lower/upper bounds of the optimization problem; and
(5) Exploiting local minima in Dynamic Voltage Scaling (DVS) to further reduce the energy consumption of HPC applications within a data center.

This paper is organized as follows. Section 2 discusses related work. Section 3 defines the Cloud computing scenario and the problem description. In Section 4, different policies for allocating applications to data centers efficiently are described. Section 5 explains the evaluation methodology and simulation setup, followed by the analysis of the performance results in Section 6. Section 7 presents the conclusion and future work.

2. Related work

Table 1 gives an overview of previous work which addresses any of the five aspects considered by this paper. To the best of our knowledge, except our work, there is no previous work which collectively addresses all five aspects.

Most previous work addresses energy-efficient computing for servers [5]. But most of them focuses on reducing energy consumption in data centers for web workloads [60,12]. Thus, they assume that energy is an increasing function of CPU frequency since web workloads have the same execution time per request. However, HPC workloads have different execution time depending on specific application requirements. Hence, the energy–CPU-frequency relationship of a HPC workload is significantly different from that of a web workload as discussed in Section 4.2. Therefore, in this paper, we define a generalized power model and adopt a more general strategy to scale up or down the CPU frequency.

Some previous work examine how energy can be saved for executing HPC applications. Bradley et al. [6] proposed algorithms to minimize the power utilization by using workload history and predicting future workload within acceptable reliability. Lawson and Smirni [39] proposed an energy saving scheme that dynamically adjusts the number of CPUs in a cluster to operate in "sleep" mode when the utilization is low. Tesauro et al. [57] presented an application of batch reinforcement learning combined with nonlinear function approximation to optimize multiple aspects of data center behavior such as performance, power, and availability.

But these solutions target to save energy within a single server or a single data center (with many servers) in a *single* location. Since our generic scheduling policy improves the energy efficiency across data centers in *multiple* locations with different carbon emission rates, it can be used in conjunction with these solutions to utilize any energy efficiency already implemented in a single location.

Table 1
Comparison of related work.

	CO_2 emission/energy consumption	HPC workload characteristic	Multiple data centers	Energy cost aware scheduling	Market-oriented schedulers
Our work	X	X	X	X	X
Bradley et al. [6]	X	X			
Lawson and Smirni [39]	X	X			
Tesauro [57]	X	X			
Orgerie et al. [45]	X	X	X		
Patel et al. [47]			X		
Chase et al. [11]				X	X
Burge et al. [9]	X	X		X	

There are some studies on energy efficiency in Grids, which comprise resource sites in multiple locations similar to our scope. Orgerie et al. [45] proposed a prediction algorithm to reduce power consumption in large-scale computational grids such as Grid5000 by aggregating the workload and turning off unused CPUs. Hence, they do not consider using DVS to save power for CPUs. Patel et al. [47] proposed allocating Grid workload on a global scale based on the energy efficiency at different data centers. But, their focus is on reducing temperature, and thus do not examine how energy consumption can be reduced by exploiting different power efficiency of CPUs, energy costs, and carbon emission rates across data centers. In addition, they do not focus on any particular workload characteristics, whereas we focus on HPC workload.

Not much previous work studies the energy sustainability issue from an economic cost perspective. To address energy usage, Chase et al. [11] adopted an economic approach to manage shared server resources in which services "bid" for resources as a function of delivered performance. Burge et al. [9] scheduled tasks to heterogeneous machines and made admission decisions based on the energy costs of each machine to maximize the profit in a single data center. But, both of them do not study the critical relationship between carbon emissions (environmental sustainability) and profit (economic sustainability) for the energy sustainability issue, and how they can affect each other. On the other hand, we examine how carbon emissions can be reduced for executing HPC applications with negligible effect on the profit of the Cloud provider.

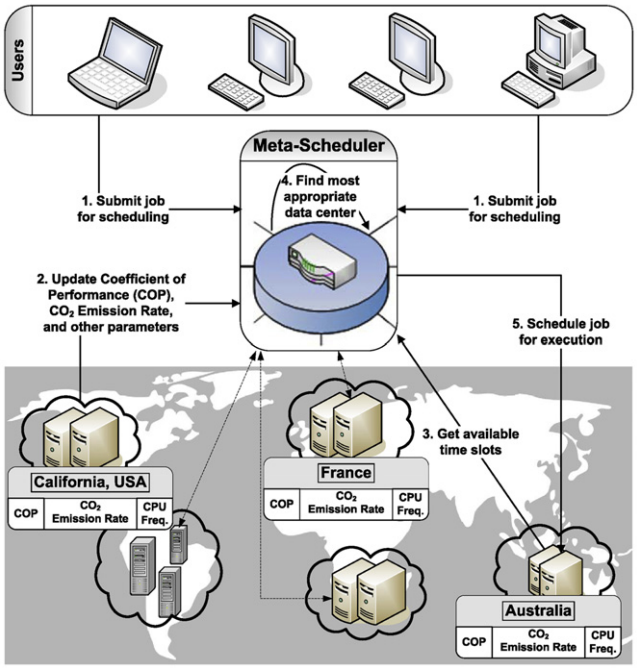


Fig. 2. Cloud meta-scheduling protocol.

3. Meta-scheduling model

3.1. System model

Our system model is based on the Cloud computing environment, whereby Cloud users are able to tap the computational power offered by the Cloud providers to execute their HPC applications. The Cloud meta-scheduler acts as an interface to the Cloud infrastructure and schedules applications on the behalf of users as shown in Fig. 2. It interprets and analyzes the service requirements of a submitted application and decides whether to accept or reject the application based on the availability of CPUs. Its objective is to schedule applications such that the carbon emissions can be reduced and the profit can be increased for the Cloud provider, while the Quality of Service (QoS) requirements of the applications are met. As data centers are located in different geographical regions, they have different carbon emission rates and energy costs depending on regional constraints. Each data center is responsible for updating this information to the meta-scheduler for energy-efficient scheduling. The two participating parties, Cloud users and Cloud providers, are discussed below with their objectives and constraints:

(1) **Cloud users:** The Cloud users need to run HPC applications/workloads which are compute-intensive with low data transfer requirements, and thus require parallel and distributed processing to significantly reduce their execution time. The users submit parallel non-moldable applications with their QoS and processing requirements to the Cloud meta-scheduler. Each application must be executed within an individual data center and does not have preemptive priority. The reason for this requirement is that the synchronization among various tasks of parallel applications can be affected by communication delays when applications are executed across multiple data centers. Furthermore, since the main aim of this paper is to design high-level application-independent meta-scheduling policies, we do not want to consider the fine-grained details of HPC workload (such as considering the impact of communication and synchronization, and their overlapping with computation), which are more applicable at the local scheduler level. The objective of the user is to have his application completed by the specified deadline. Deadlines are hard, i.e. the user will benefit from the HPC resources only if the application completes before its deadline [49].

To facilitate the comparison of various policies described in this work, the estimated execution time of an application is assumed to be known by the user at the time of submission [24]. This can be derived based on user-supplied information, experimental data, application profiling or benchmarking, and other techniques. Existing performance prediction techniques (based on analytical modeling [44], empirical [4] and historical data [55,37,53]) can also be applied to estimate the execution time of parallel applications. However, in reality, it is

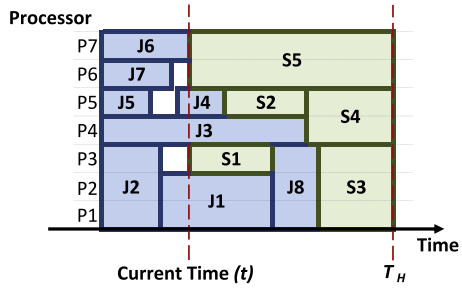


Fig. 3. Free time slots.

not always possible to estimate the execution time of an application accurately. But, in Cloud computing where users pay based on actual resource usage, a user will have to pay more than expected if the execution time of his application has been under-estimated. Thus, a user must still be given the privilege of whether to accept or change any automatically derived estimate before submission.

(2) **Cloud providers:** A Cloud provider has multiple data centers distributed across the world. For example, Amazon [1] has data centers in many cities across Asia, Europe, and United States. Each data center has a local scheduler that manages the execution of incoming applications. The Cloud meta-scheduler interacts with these local schedulers for application execution. Each local scheduler periodically supplies information about available time slots (t_s, t_e, n) to the meta-scheduler, where t_s and t_e are the start time and end time of the slot respectively and n is the number of CPUs available for the slot. Within a data center, the free time slots are obtained based on the approach used by Singh et al. [54]. The CPU availability at particular times in the future is maintained by the local scheduler. The free time slot information is generated until a given time horizon, thus creating windows of availability or free time slots; the end time of a free time slot is either the end time of a job in the waiting queue or the planning horizon. The time horizon is set to ∞, thus all the free time slot information is disclosed to the meta-scheduler. An example is given in Fig. 3, where S1, S2, …, S5 are free time slots. The Cloud Provider also supplies execution price and I/O data transfer cost to the meta-scheduler.

To facilitate energy-efficient computing, each local scheduler also supplies information about the carbon emission rate, Coefficient of Performance (COP), electricity price, CPU power–frequency relationship, Million Instructions Per Second (MIPS) rating of CPUs at the maximum frequency, and CPU operating frequency range of the data center. The MIPS rating is used to indicate the overall performance of a CPU. All CPUs within a data center are homogeneous, but CPUs can be heterogeneous across data centers. The carbon emission rates are calculated based on the fuel type used in electric power generation. These are published regularly by various government agencies such as US Energy Information Administration (EIA). COP of the data center's cooling system is defined as the amount of cooling delivered per unit of electrical power consumed. COP can be measured by monitoring the energy consumption by various components of the cooling system [46]. Various parameters of CPUs at the data center can be derived experimentally [29].

3.2. Data center energy model

The major contributors for the total energy usage in a data center are IT equipment (which consists of servers, storage devices, and network equipment) and cooling systems [19]. Other systems such as lighting are not considered due to their negligible contribution to the total energy usage.

Within a data center, the total energy usage of a server depends on its CPUs, memory, disks, fans, and other components [22]. However, for simplicity, we only compute the energy usage of a server based on its CPUs due to two reasons. First, as pointed out by Fan et al. [22], the energy usage of the server varies depending on the type of workload executed. Since we only consider compute-intensive HPC applications and the CPUs use the largest proportion of energy in a server, it is sufficient in our case to only model CPU energy usage. Second, the aim of this paper is to examine how a meta-scheduler can achieve energy efficiency on a global scale by exploiting the heterogeneity across multiple data centers. Hence we do not focus in detail on how to save energy usage locally within a data center, whereby energy can potentially be saved through other components in the server, such as memory and disks. But, our proposed meta-scheduling policies can easily compliment any other solutions that focus on saving energy within a data center in this regard.

The power consumption of a CPU can be reduced by lowering its supply voltage using DVS. DVS is an efficient way to manage dynamic power dissipation during computation. The power consumption model of CPUs which are generally composed of CMOS circuits is given by: $P = \alpha V^2 f + I_{leak} V + P_{short}$, where P is the power dissipation, V is the supply voltage, f is the clock frequency, I_{leak} is the leakage current, and P_{short} is the short circuit power dissipated during the voltage switching process [8,48]. The first term constitutes the dynamic power of the CPU and the second term constitutes the static power. P_{short} is generally negligible in comparison to other terms.

Since the voltage can be expressed as a linear function of frequency in the CMOS logic, the power consumption P_i of a CPU in a data center i is approximated by the following function (similar to previous work [60,12]): $P_i = \beta_i + \alpha_i f^3$, where β_i is the static power consumed by the CPU, α_i is the proportionality constant, and f is the frequency at which the CPU is operating. We use this cubic relationship between the operating frequency and power consumption since this paper focuses on compute-intensive workload and to the best of our knowledge, the cubic relationship is the most commonly used metric for CPU power in previous work [60,12]. We also consider that a CPU of data center i can adjust its frequency from a minimum of f_i^{min} to a maximum of f_i^{max} discretely. The frequency levels supported by a CPU typically varies for different CPU architecture. For instance, Intel Pentium M 1.6 GHz CPU supports 6 V from 0.956 V to 1.484 V.

The energy cost of the cooling system depends on its COP [42,56]. COP is an indication for the efficiency of the cooling system, which is defined as the ratio of the amount of energy consumed by CPUs to the energy consumed by the cooling system. However, COP is not constant and varies with the cooling air temperature. We assume that COP will remain constant during a scheduling cycle and data centers will update the meta-scheduler whenever COP changes. Thus, the total energy consumed by the cooling system in a data center i is given by:

$$E_i^h = \frac{E_i^c}{COP_i} \tag{1}$$

where E_i^c is the total energy consumed by CPUs and E_i^h is the total energy consumed by cooling devices.

The total energy consumed by data center i can then be approximated by:

$$E_i^{total} = E_i^c + E_i^h = \left(1 + \frac{1}{COP_i}\right) E_i^c = \left(\frac{COP_i + 1}{COP_i}\right) E_i^c.$$

Therefore, the data center efficiency (DCiE) [59] is given as:

$$DCiE = \frac{COP}{COP + 1}.$$

3.3. Relation between execution time and CPU frequency

Since we use DVS to scale up/down the CPU frequency, the execution time of an application can significantly vary according

Table 2
Parameters of a data center i.

Parameter	Notation
Carbon emission rate (kg/kW h)	$r_i^{CO_2}$
Average COP	COP_i
Electricity price ($/kW h)	p_i^e
Data transfer price ($/GB) for upload/download	p_i^{DT}
CPU power	$P_i = \beta_i + \alpha_i f^3$
CPU frequency range	$[f_i^{min}, f_i^{max}]$
Time slots (start time, end time, number of CPUs)	(t_s, t_e, n)

to the CPU frequency. However, the decrease in execution time due to the increase in CPU frequency depends on whether the application is CPU bound or not. For example, if the performance of an application is completely dependent on the CPU frequency, then its execution time will be inversely proportional to the change in CPU frequency. Thus, the execution time of an application is modeled according to the definition proposed by Hsu et al. [34]:

$$T(f) = T(f^{max}) \times \left(\gamma^{cpu}\left(\frac{f^{max}}{f} - 1\right) + 1\right) \tag{2}$$

where $T(f)$ is the execution time of the application at CPU frequency f, $T(f^{max})$ is the execution time of the application at the maximum CPU frequency f^{max}, and γ^{cpu} is the CPU boundness of the application.

If the value of γ^{cpu} decreases, the CPU boundness of the application will also decrease, which results in potentially more energy reduction by using DVS in servers within a data center (see Section 4.2). It is however important to note that the CPU boundness of an application varies based on the CPU architecture, as well as memory and disks. Like many prior studies [20,28], we still use this factor to model the CPU usage intensity of an application in a simple generic manner so as to optimize its energy consumption accordingly. However, for all our experiments, we have used the worst case value of $\gamma^{cpu} = 1$ to analyze the performance of our heuristics.

3.4. Problem description

Let a Cloud provider have N data centers distributed in different locations, and J is the number of applications currently executing on these data centers. All the parameters associated with a data center i are given in Table 2. A data center i incurs carbon emission based on its carbon emission rate $r_i^{CO_2}$ (kg/kW h). To execute an application, the Cloud provider has to pay data center i the energy cost and data transfer cost depending on its electricity price p_i^e ($/kW h) and data transfer price p_i^{DT} ($/GB) for upload/download respectively. The Cloud provider then charges fixed prices to the user for executing his application based on the CPU execution price p^c ($/CPU/hour) and data transfer price p^{DTU} ($/GB) for the processing time and upload/download respectively.

A user submits his requirements for an application j in the form of a tuple $(d_j, n_j, e_{j1}, \ldots, e_{jN}, \gamma_j^{cpu}, (DT)_j)$, where d_j is the deadline to complete application j, n_j is the number of CPUs required for application execution, e_{ji} is the application execution time on the data center i when operating at the maximum CPU frequency, γ_j^{cpu} is the CPU boundness of the application, and $(DT)_j$ is the size of data to be transferred. For simplicity, we assume that users are able to specify their processing requirements (n_j, e_{ji}, and γ_j^{cpu}). It is realistic for a user to specify n_j and e_{ji} in an utility computing environment since the user need to pay for usage. For instance, in Amazon EC2 [1], the user can buy one Compute unit with the number of processors but on hourly basis. Thus, even if the user's application finishes before one hour, the user need to pay for one complete hour.

In addition, let f_{ij} be the initial frequency at which CPUs of a data center i operate while executing application j. Hence executing application j on data center i results in the following:

(i) Energy consumption of CPUs

$$E_{ij}^c = (\beta_i + \alpha_i (f_{ij})^3) \times n_j e_{ji} \times \left(\gamma_j^{cpu}\left(\frac{f_i^{max}}{f_{ij}} - 1\right) + 1\right). \tag{3}$$

(ii) Total energy which consist of the cooling system and CPUs

$$E_{ij} = \frac{COP_i + 1}{COP_i} \times E_{ij}^c. \tag{4}$$

(iii) Energy cost

$$C_{ij}^e = p_i^e \times E_{ij}. \tag{5}$$

(iv) Carbon emission

$$(CO_2E)_{ij} = r_i^{CO_2} \times E_{ij}. \tag{6}$$

(v) Execution Profit

$$(ProfExec)_{ij} = n_j e_{ji} p^c - C_{ij}^e. \tag{7}$$

(vi) Data Transfer Profit

$$(ProfData)_{ij} = (DT)_j \times (p^{DTU} - p_i^{DT}). \tag{8}$$

(vii) Profit

$$(Prof)_{ij} = (ProfExec)_{ij} + (ProfData)_{ij}. \tag{9}$$

The carbon emission $(CO_2E)_{ij}$ (Eq. (6)) incurred by application j is computed using the carbon emission rate $r_i^{CO_2}$ of data center i. However, this means that $(CO_2E)_{ij}$ only reflects the average carbon emission incurred since $r_i^{CO_2}$ is an average rate. We can only use $r_i^{CO_2}$ since the exact amount of carbon emission produced depends on the type of fuel used to generate the electricity and no detailed data is available in this regard.

The profit $(Prof)_{ij}$ (Eq. (9)) gained by the Cloud provider from the execution of an application j on data center i includes the execution profit $(ProfExec)_{ij}$ and input/output data transfer profit $(ProfData)_{ij}$. Studies [27,3] have shown that the ongoing operational costs (such as energy cost) of data centers greatly surpass their one-time capital costs (such as hardware and support infrastructure costs). Hence, when computing the execution profit $(ProfExec)_{ij}$, we assume that the CPU execution price p^c charged by the Cloud provider to the user already includes the one-time capital costs of data centers, so that we only subtract the ongoing energy cost C_{ij}^e of executing applications from the revenue. The data transfer profit $(ProfData)_{ij}$ is the difference between the cost paid by the user to the provider and the cost incurred for transferring the data to the data center.

The meta-scheduling problem can then be formulated as:

$$\text{Minimize Carbon Emission} = \sum_i^N \sum_j^J x_{ij} (CO_2E)_{ij} \tag{10}$$

$$\text{Maximize Profit} = \sum_i^N \sum_j^J x_{ij} (Prof)_{ij}. \tag{11}$$

Subject to:

(a) Response time of application $j < d_j$
(b) $f_i^{min} < f_{ij} < f_i^{max}$
(c) $\sum_i^N x_{ij} \leq 1$
(d)

$$x_{ij} = \begin{cases} 1 & \text{if application } j \text{ allocated to data center } i \\ 0 & \text{otherwise.} \end{cases}$$

The dual objective functions (10) and (11) of the meta-scheduling problem are to minimize the carbon emission and maximize the profit of a Cloud provider. Constraint (a) ensures that the deadline requirement of an application is met. But it is difficult to

calculate the exact response time of an application since applications have different sizes, require multiple CPUs, and have very dynamic arrival rates [12]. Moreover, this problem maps to the 2-dimensional bin-packing problem which is NP-hard in nature [41] (see Appendix A for the proof). Hence we propose various scheduling policies to heuristically approximate the optimum.

4. Meta-scheduling policies

The meta-scheduler periodically assigns applications to data centers at a fixed time interval called the *scheduling cycle*. This enables the meta-scheduler to potentially make a better selection choice of applications when mapping from a larger pool of applications to the data centers, as compared to during each submission of an application. In each scheduling cycle, the meta-scheduler collects the information from both data centers and users.

In general, a meta-scheduling policy consists of two phases: (1) mapping phase, in which the meta-scheduler first maps an application to a data center; and (2) scheduling phase, in which the scheduling of applications is done within the data center, where the required time slots is chosen to execute the application. Depending on the objective of Cloud provider whether to minimize carbon emission or maximize profit, we have designed various mapping policies which are discussed in the subsequent section. To further reduce the energy consumption within the data center, we have designed a DVS based scheduling policy for the local scheduler of a data center.

4.1. Mapping phase (across many data centers)

We have designed the following meta-scheduling policies to map applications to data centers depending on the objective of the Cloud provider:

4.1.1. Minimizing carbon emission

The following policies optimize the global carbon emission of all data centers while keeping the number of deadline misses low.

- **Greedy Minimum Carbon Emission (GMCE):** Since the aim is to minimize the carbon emission across all the data centers, we want the most number of applications to be executed on data centers with the least carbon emission. Hence applications are sorted by their deadline (earliest first) to reduce the deadline misses, while data centers are sorted by their carbon emission (lowest first), which is computed as: $r_i^{CO_2} \times \frac{COP_i+1}{COP_i} \times (\beta_i + \alpha_i (f_i^{max})^3)$. Each application is then mapped to a data center in this ordering.
- **Minimum-Carbon-Emission–Minimum-Carbon-Emission (MCE–MCE):** MCE–MCE is based on the Min–Min heuristic [35] which has performed very well in previous studies of different environments [7]. The meta-scheduler first finds the "best" data center for all applications that are considered. Then among these application–data center pairs, the meta-scheduler selects the "best" pair to map first. Since the aim is to minimize the carbon emission, the "best" pair has the minimum carbon emission $(CO_2E)_{ij}$, i.e. minimum fitness value of executing application j on data center i. MCE–MCE has the following steps:
  - Step 1: For each application in the list of applications to be mapped, find the data center of which the carbon emission is the minimum, i.e. minimum $(CO_2E)_{ij}$ (the first MCE), among all data centers which can complete the application by its deadline. If there is no data center where the application can be completed by its deadline, the application is removed from the list of applications to be mapped.
  - Step 2: Among all the application–data center pairs found in Step 1, find the pair that results in the minimum carbon emission, i.e. minimum $(CO_2E)_{ij}$ (the second MCE). Then, map the application to the data center, and remove it from the list of applications to be mapped.
  - Step 3: Update the available time slots from data centers.
  - Step 4: Do Step 1 to 3 again until all applications are mapped.

4.1.2. Maximizing profit

The following policies optimize the global profit of all data centers while keeping the number of deadline misses low.

- **Greedy Maximum Profit (GMP):** Since the aim is to maximize the profit across all the data centers, we want the most number of applications to be executed on data centers with the least energy cost. Hence applications are sorted by their deadline (earliest first) to reduce the deadline misses, while data centers are sorted by their energy cost (lowest first), which is computed as: $p_i^e \times \frac{COP_i+1}{COP_i} \times (\beta_i + \alpha_i (f_i^{max})^3)$. Each application is then mapped to a data center in this ordering.
- **Maximum-Profit–Maximum-Profit (MP–MP):** MP–MP works in the same way as MCE–MCE. However, since the aim is to maximize the profit, the "best" pair has the maximum profit $(Prof)_{ij}$, i.e. maximum fitness value of executing application j on data center i. Hence the steps of MP–MP are the same as MCE–MCE, except the following differences:
  - Step 1: For each application in the list of applications to be mapped, find the data center of which the profit is the maximum, i.e. maximum $(Prof)_{ij}$ (the first MP), among all data centers which can complete the application by its deadline.
  - Step 2: Among all the application–data center pairs found in Step 1, find the pair that results in the maximum profit, i.e. maximum $(Prof)_{ij}$ (the second MP).

4.1.3. Minimizing carbon emission and maximizing profit (MCE–MP)

MCE–MP works in the same way as MCE–MCE. But, since the aim is to minimize the total carbon emission while maximizing the total profit across all the data centers, MCE–MP handles the trade-off between carbon emission and profit which may be conflicting. Hence the steps of MCE–MP are the same as MCE–MCE, except the following differences:

- Step 1: For each application in the list of applications to be mapped, find the data center of which the carbon emission is the minimum, i.e. minimum $(CO_2E)_{ij}$ (the first MCE), among all data centers which can complete the application by its deadline.
- Step 2: Among all the application–data center pairs found in Step 1, find the pair that results in the maximum profit, i.e. maximum $(Prof)_{ij}$ (the second MP).

One more mapping policy can be designed to minimize carbon emission and maximize profit simultaneously by reversing the above two steps. This policy is named as MP–MCE (Maximizing Profit and Minimizing Carbon Emission).

4.2. Scheduling phase (within a data center)

The energy consumption and carbon emission are further reduced within a data center by using DVS at the CPU level to save energy by scaling down the CPU frequency. Thus, before the meta-scheduler assigns an application to a data center, it decides the time slot in which the application should be executed and the frequency at which the CPU should operate to save energy. But, since a lower CPU frequency can increase the number of applications rejected due to the deadline misses, the scheduling of

applications within the data center can be of two types: (1) CPUs run at the maximum frequency (i.e. without DVS) or (2) CPUs run at various frequencies using DVS (i.e. with DVS). It is important to adjust DVS appropriately in order to reduce the number of deadline misses and energy consumption simultaneously.

The meta-scheduler will first try to operate the CPU at a frequency in the range $[f_i^{min}, f_i^{max}]$ nearest to the optimal CPU frequency $f_i^{opt} = \sqrt[3]{\frac{\beta_i}{2\alpha_i}}$ for $\gamma^{cpu} = 1$ (see Appendix B for the analysis of exploiting local minima in DVS). Since the CPU frequency of data center i can only operate in the interval $[f_i^{min}, f_i^{max}]$, we define $f_i^{opt} = f_i^{min}$ if $f_i^{opt} < f_i^{min}$, and $f_i^{opt} = f_i^{max}$ if $f_i^{opt} > f_i^{max}$.

If the deadline of an application will be violated, the meta-scheduler will scale up the CPU frequency to the next level and then try again to find the free time slots to execute the application. If the meta-scheduler fails to schedule the application on the data center as no free time slot is available, then the application is forwarded to the next data center for scheduling (the ordering of data centers depends on various policies as described in Section 4.1).

4.3. Lower bound and upper bound

Due to the NP hardness of the meta-scheduling problem described in Section 3.4, it is difficult to find the optimal profit and carbon emission in polynomial time. Thus, to estimate the performance of our scheduling algorithms, we present a lower bound for the carbon emission and an upper bound for the profit of the Cloud provider respectively. Both bounds are derived based on the principle that we can get the minimum carbon emission or the maximum profit when most of the applications are executed on the most "efficient" data center and also at the optimal CPU frequency. For carbon emission minimization, the most "efficient" data center incurs the minimum carbon emission for executing applications, while for profit maximization, the most "efficient" data center results in the minimum energy cost.

For the sole purpose of deriving the lower and upper bounds, we relax three constraints of our system model so as to map the maximal number of applications to the most "efficient" data center. First, we relax the constraint that when an application is executed at the maximum CPU frequency, it will result in the maximum energy consumption. Instead, we assume that even though all applications are executed at the maximum CPU frequency, the actual energy consumed by them for calculating their carbon emission still remains at the optimal CPU frequency using DVS. Second, although the applications considered in the system model are parallel applications with fixed CPU requirements, we relax this constraint to applications that are moldable in the required number of CPUs. Thus, the runtime of applications will decrease linearly when it is scheduled on a larger number of CPUs. This in turn increases the number of applications that can be allocated to the most "efficient" data center with the minimum energy possible. Third, the applications in the system model are arriving dynamically in many different scheduling cycles, but for deriving the bounds, all applications are considered in only one scheduling cycle and mapped to data centers. This forms the best ideal bound scenario, where all the incoming applications are known in advance. Hence the actual dynamic scenario definitely has worse performance than that of the ideal bound scenario.

It is important to note that the bounds of carbon emission and profit obtained with these three assumptions are unreachable loose bounds of the system model. This is because data centers will be executing the maximum possible workload with 100% utilization of their CPUs, while the least possible energy consumption is still considered for the purpose of comparison.

Let TWL be the total workload scheduled, TCE be the total carbon emission, and TP be the total profit. The lower bound for the carbon emission is derived through the following steps:

Step 1: Applications are sorted by their deadline (earliest first) to reduce the deadline misses, while data centers are sorted by their carbon emission (lowest first), which is computed as: $r_i^{CO_2} \times \frac{COP_i+1}{COP_i} \times (\beta_i + \alpha_i (f_i^{max})^3)$. Each application is then mapped to a data center in this ordering.

Step 2: For each application j, search for a data center i, starting from the most "efficient" one, where the application j can be scheduled without missing its deadline when running at the maximum CPU frequency.

Step 3: If a data center i is not found, then application j will be removed from the list of potential applications. Go to Step 2 to schedule other applications.

Step 4: If a data center i is found, application j is assigned to it and molded such that there is no fragmentation in the schedule of data center i for executing applications.

Step 5: $TWL+ = n_j \times e_{ji}$

Step 6: $TCE+ = r_i^{CO_2} \times \frac{COP_i+1}{COP_i} \times$ (Power consumption of the CPU at optimal CPU frequency) $\times n_j \times$ (Execution time of application j at optimal CPU frequency)

Step 7: $TP+ = (1 - (p_i^e \times \frac{COP_i+1}{COP_i} \times$ (Power consumption of the CPU at optimal CPU frequency))) $\times n_j \times$ (Execution time of application j at optimal CPU frequency).

Step 8: Repeat from Step 2 until all applications are scheduled.

$\frac{TCE}{TWL}$ will be the lower bound of the average carbon emission due to the execution of all applications across multiple data centers of the Cloud provider.

To derive the upper bound for the profit, the steps remain the same, except the following differences:

- In Step 1, data centers are sorted by their energy cost (lowest first), which is computed as: $p_i^e \times \frac{COP_i+1}{COP_i} \times (\beta_i + \alpha_i (f_i^{max})^3)$.
- $\frac{TP}{TWL}$ will be the upper bound of the average profit.

5. Performance evaluation

Configuration of applications: We use workload traces from Feitelson's Parallel Workload Archive (PWA) [23] to model the HPC workload. Since this paper focuses on studying the requirements of Cloud users with HPC applications, the PWA meets our objective by providing workload traces that reflect the characteristics of real parallel applications. Our experiments utilize the first week of the LLNL Thunder trace (January 2007 to June 2007). The LLNL Thunder trace from the Lawrence Livermore National Laboratory (LLNL) in USA is chosen due to its highest resource utilization of 87.6% among available traces to ideally model a heavy workload scenario. From this trace, we obtain the submit time, requested number of CPUs, and actual runtime of applications. We set the CPU boundness of all workload as 1 (i.e. $\gamma^{cpu} = 1$) to examine the worst case scenario of CPU energy usage. We use a methodology proposed by Irwin et al. [36] to synthetically assign deadlines through two classes namely Low Urgency (LU) and High Urgency (HU).

An application j in the LU class has a high ratio of $deadline_j / runtime_j$ so that its deadline is definitely longer than its required runtime. Conversely, an application j in the HU class has a deadline of low ratio. Values are normally distributed within each of the high and low deadline parameters. The ratio of the deadline parameter's high-value mean and low-value mean is thus known as the high:low ratio. In our experiments, the deadline high:low ratio is 3, while the low-value deadline mean and variance is 4 and 2 respectively. In other words, LU applications have a high-value deadline mean of 12, which is 3 times longer than HU applications with a low-value deadline mean of 4. The arrival sequence of applications from the HU and LU classes is randomly distributed.

Configuration of data centers: We model 8 data centers with different configurations as listed in Table 3. Carbon emission rates

Table 3
Characteristics of data centers.

Location	Carbon emission rate[a] (kg/kW h)	Electricity price[b] ($/kW h)	CPU power factors		CPU frequency level		Number of CPUs
			β	α	f_i^{max}	f_i^{opt}	
New York, USA	0.389	0.15	65	7.5	1.8	1.630324	2050
Pennsylvania, USA	0.574	0.09	75	5	1.8	1.8	2600
California, USA	0.275	0.13	60	60	2.4	0.793701	650
Ohio, USA	0.817	0.09	75	5.2	2.4	1.93201	540
North Carolina, USA	0.563	0.07	90	4.5	3.0	2.154435	600
Texas, USA	0.664	0.1	105	6.5	3.0	2.00639	350
France	0.083	0.17	90	4.0	3.2	2.240702	200
Australia	0.924	0.11	105	4.4	3.2	2.285084	250

[a] Carbon emission rates are derived from a US Department of Energy (DOE) document (Appendix F-Electricity Emission Factors 2007) [14].
[b] Electricity prices are average commercial prices till 2007 based on a US Energy Information Administration (EIA) report [16].

and electricity prices at various data center locations are averages over the entire region and derived from the data published by US Department of Energy [14] and Energy Information Administration [16].

We derived the power related parameter, from a recent work presented by Wang and Lu [60] who also focus on the similar problem of considering the energy consumption of heterogeneous CPUs. They used the experimental data from the work by Rusu et al. [51], 'Desktop CPU Power Survey' by Silent PC Review [13], and 'CPU Performance Charts' by Tom Hardware [33] to estimate various server parameters. The CPU Performance Charts [33] provides the comprehensive comparison of AMD and Intel processors using more than 20 benchmarks such as audio and video encoding tools and multitasking applications. Similar empirical data with experimental details are given in Desktop CPU Power Survey [13] for power and energy efficiency of various Intel and AMD processors, such as Athlon 64 3000+ and Pentium 4 630, at idle and full load power. Thus, the power parameters (i.e. CPU power factors and frequency level) of the CPUs at different data centers are derived based on these experimental data [60]. The values of α and β are set such that the ratio of static power and dynamic power can cover a wide variety of CPUs.

Current commercial CPUs only support discrete frequency levels, such as the Intel Pentium M 1.6 GHz CPU which supports 6 V levels. We consider discrete CPU frequencies with 5 levels in the range $[f_i^{min}, f_i^{max}]$. For the lowest frequency f_i^{min}, we use the same value used by Wang and Lu [60], i.e. f_i^{min} is 37.5% of f_i^{max}.

To increase the utilization of data centers and reduce the fragmentation in the scheduling of parallel applications, the local scheduler at each data center uses Conservative Backfilling with advance reservation support as proposed by Mu'alem and Feitelson [43]. The meta-scheduler schedules applications periodically at each scheduling cycle of 50 s, which is to ensure that the meta-scheduler can receive at least one application in every scheduling cycle.

The COP (power usage efficiency) value of data centers is randomly generated using a uniform distribution between [0.6, 3.5] as indicated in the study conducted by Greenberg et al. [32]. To avoid the energy cost of a data center exceeding the revenue generated by the Cloud provider, the CPU execution price charged by the provider to the user is fixed at $0.40 /CPU/h which is approximately twice of the maximum energy cost at a data center.

Performance metrics: We observe the performance from both user and provider perspectives. From the provider perspective, four metrics are necessary to compare the policies: average energy consumption, average carbon emission, profit gained, and workload executed. The *average energy consumption* compares the amount of energy saved by using different scheduling algorithms, whereas the *average carbon emission* compares its corresponding environmental impact. Since minimizing the carbon emission can affect a Cloud provider economically by decreasing its profit, we have considered the *profit gained* as another metric to compare different algorithms. It is important to know the effect of various meta-scheduling policies on energy consumption, since higher energy consumption is likely to generate more carbon emission for worse environmental impact and incur more energy cost for operating data centers.

From the user perspective, we observe the performance of varying: (1) urgency class and (2) arrival rate of applications. For the urgency class, we use various percentages (0%, 20%, 40%, 60%, 80%, and 100%) of HU applications. For instance, if the percentage of HU applications is 20%, then the percentage of LU applications is the remaining 80%. For the arrival rate, we use various factors (10 (low), 100 (medium), 1000 (high), and 10 000 (very high)) of submit time from the trace. For example, a factor of 10 means an application with a submit time of 10 s from the trace now has a simulated submit time of 1 s. Hence, a higher factor represents higher workload by shortening the submit time of applications.

Experimental scenarios: To comprehensively evaluate the performance of our algorithms, we examine various experimental scenarios that are classified as:

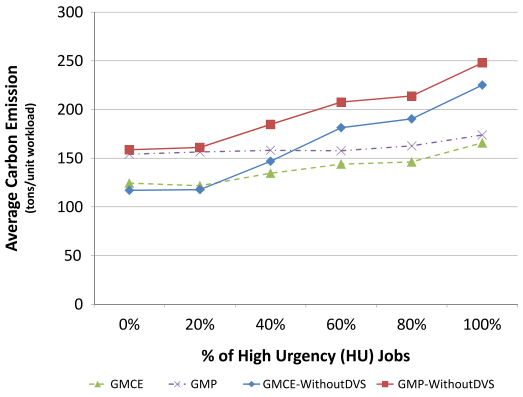
(a) **Evaluation without data transfer cost** (Section 6.1): In the first set of experiments (Section 6.1.1), we evaluate the importance of our mapping policies which consider global factors such as carbon emission rate, electricity price and data center efficiency. In these experiments, we also evaluate the effectiveness of exploiting local minima in our local DVS policy (Sections 6.1.1 and 6.1.2). Then, in the next set of experiments (Section 6.1.3), we compare the performance of our proposed algorithms with the lower bound (carbon emission) and the upper bound (profit).

To evaluate the overall best among the proposed algorithms, we conduct more experiments by varying different factors which can affect their performance:
- Impact of urgency and arrival rate of applications (Section 6.1.4)
- Impact of carbon emission rate (Section 6.1.5)
- Impact of electricity price (Section 6.1.6)
- Impact of data center efficiency (Section 6.1.7).

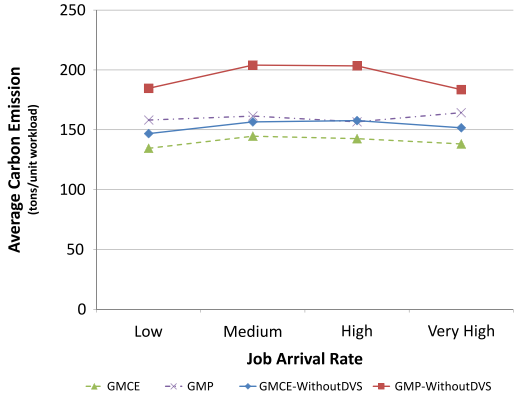
(b) **Evaluation with data transfer cost** (Section 6.2): In the last set of experiments (Section 6.2.1), we examine how the data transfer cost affect the performance of our algorithms.

6. Analysis of results

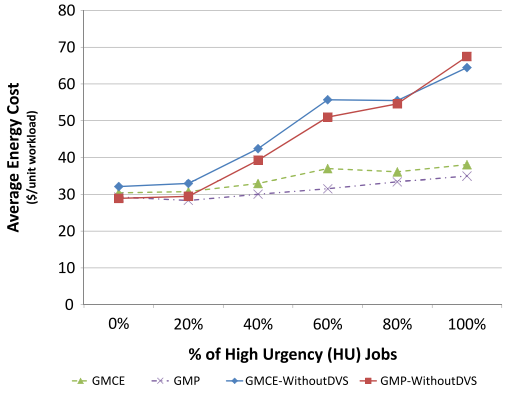
This section presents the evaluation of our proposed mapping scheduling policies based on various metrics such as carbon emission, and profit. During experimentation, the performance of MP–MCE policy in terms of carbon emission and profit was observed to be very similar to GMP with no additional benefits. Hence, to save space, the results for MP–MCE are not presented in the paper.



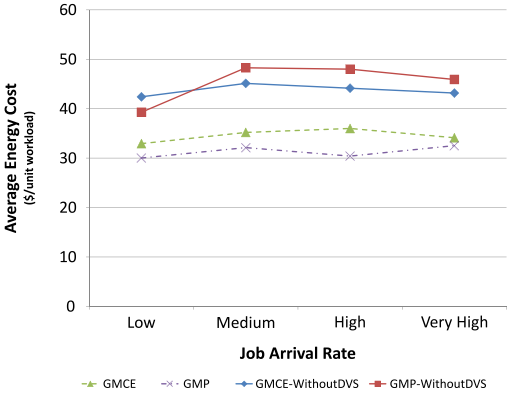
(a) Carbon emission vs. urgency.



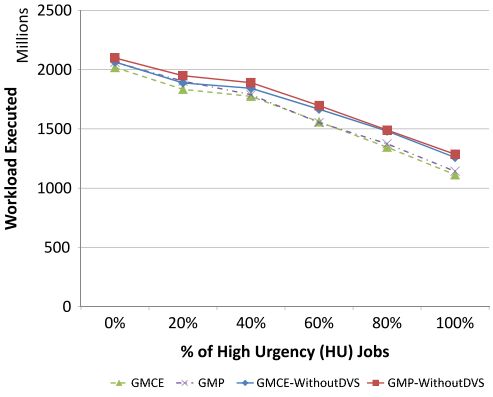
(b) Carbon emission vs. arrival rate.



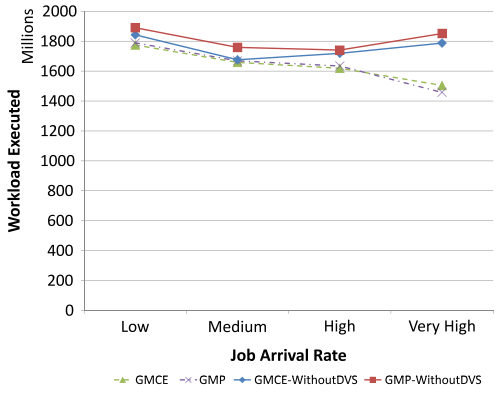
(c) Energy cost vs. urgency.



(d) Energy cost vs. arrival rate.



(e) Workload executed vs. urgency.



(f) Workload executed vs. arrival rate.

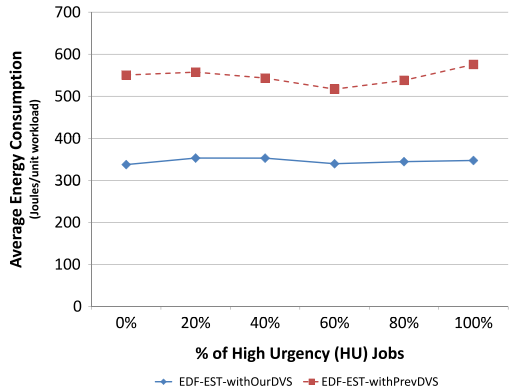
Fig. 4. Effect of mapping policy and DVS.

6.1. Evaluation without data transfer cost

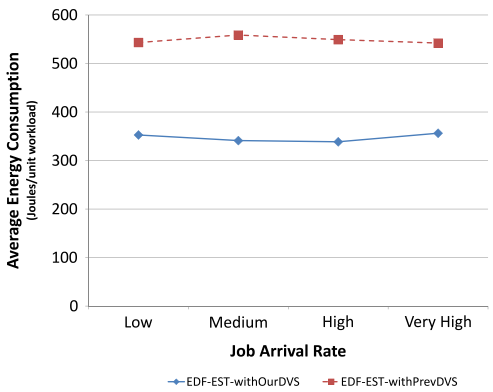
6.1.1. Effect of mapping policy and DVS

As discussed in Section 4, our meta-scheduling policies are designed to save energy at two phases, first at the mapping phase and then at the scheduling phase. Hence, in this section, we examine the importance of each phase in saving energy. These experiments also answer the question why we require special energy saving schemes at two phases.

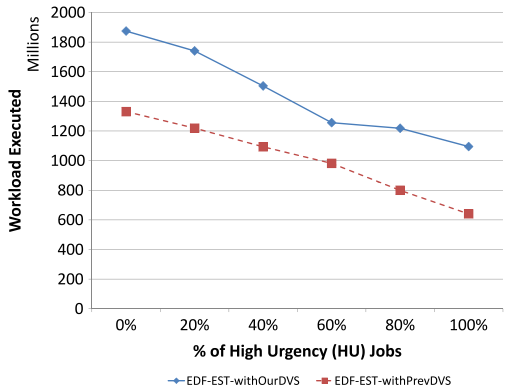
First, we examine the importance of considering the global factors at the mapping phase by comparing meta-scheduling policies without the energy saving feature at the local scheduling phase, i.e. DVS is not available at the local scheduler. Hence, we name the without DVS version of the carbon emission based policy (GMCE) and profit based policy (GMP) as GMCE-WithoutDVS and GMP-WithoutDVS respectively. The results in Fig. 4 shows that the consideration of various global factors cannot only decrease the carbon emission, but also decrease the overall energy consumption. For various urgency of applications (Fig. 4(a)), GMCE-WithoutDVS can prevent up to 10% carbon emission over GMP-WithoutDVS. For various arrival rate of applications (Fig. 4(b)), GMCE-WithoutDVS can produce up to 23% less carbon emission than GMP-WithoutDVS. The corresponding difference in energy cost (Fig. 4(c) and (d)) between them is very little (about 0%–6%). This is because with the decrease in energy consumption due to the execution of HPC workload, both carbon emission and energy cost will automatically decrease. This trend still remains by comparing GMCE and GMP, both of which uses DVS at the scheduling phase.



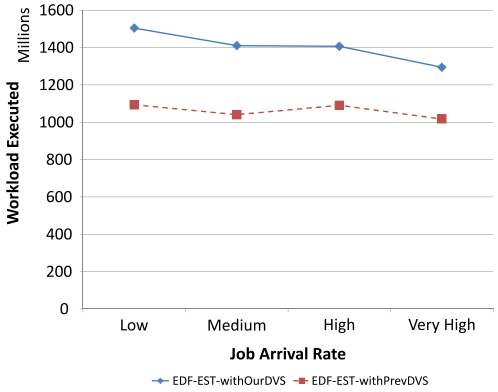
(a) Energy consumption vs. urgency.



(b) Energy consumption vs. arrival rate.



(c) Workload executed vs. urgency.



(d) Workload executed vs. arrival rate.

Fig. 5. Exploiting local minima in DVS.

Next, we examine the impact of the scheduling phase on energy consumption by comparing meta-scheduling policies with DVS (GMCE and GMP) and without DVS (GMCE-WithoutDVS and GMP-WithoutDVS). With DVS, the energy cost (Fig. 4(c)) to execute HPC workload has been reduced on average by 33% when we compare GMP with GMP-withoutDVS. With the increase in HU applications, the gap is increasing and we can get almost 50% decrease in energy cost as shown in Fig. 4(c). With the increase in arrival rate, we get a consistent 25% gain in energy cost by using DVS (Fig. 4(d)). The carbon emission is also reduced further on average by 13% with the increase in urgent applications as shown in Fig. 4(a). With the increase in arrival rate, the HPC workload executed is decreasing in the case of policies using DVS as can be observed from Fig. 4(f). This is because the execution of applications at lower CPU frequency results in more rejection of urgent applications when the arrival rate is high. Thus, HPC workload executed in the case of policies without DVS is almost the same even when the arrival rate is very high.

Finally, we examine the overall trend of change in carbon emission, workload and energy cost with respect to the number of urgent applications and job arrival rate. In Fig. 4, with the increase in the number of urgent applications, the energy cost and carbon emission is increasing while the amount of workload executed is decreasing. This is due to more applications being scheduled on less carbon-efficient sites in order to avoid missing of the deadlines. This is also the reason for all four policies to execute decreasing workload as the number of HU applications increases. Due to increase in the number of urgent applications, many applications missed the deadline, and thus got rejected. Due to cubic relationship between energy and CPU frequency, the carbon emission and energy cost is increased more rapidly in comparison to the workload which is decreasing. With respect to job arrival rate, the change in the energy cost, carbon emission and workload is very low. This is because the execution of jobs at lower CPU frequency results in more rejection of urgent jobs when the arrival rate is high.

6.1.2. Exploiting local minima in DVS

We want to highlight the importance of exploiting local minima in the DVS function while scheduling within a data center. But, to correctly highlight the difference in DVS performance for the scheduling phase of the meta-scheduling policy, we need an independent policy (which is not linked to our proposed polices) for the mapping phase. Hence, we use EDF–EST, where the applications are ordered based on Earliest Deadline First (EDF), while the data centers are ordered based on Earliest Start Time (EST). We name our proposed DVS as EDF–EST-withOurDVS that exploits the local minima in the DVS function. Our proposed DVS is compared to a previously proposed DVS named as EDF–EST-withPrevDVS, in which the CPU frequency is scaled up linearly between $[f^{min}, f^{max}]$ [60,38].

Fig. 5 shows that EDF–EST-withOurDVS has not only outperformed EDF–EST-withPrevDVS by saving about 35% of energy, but also executed about 30% more workload. This is because EDF–EST-withPrevDVS tries to run applications at the minimum CPU frequency f^{min} which may not be the optimal frequency. As discussed in Appendix B and shown in Fig. B.1, it is clear that an application executed at f^{min} may not lead to the least energy consumption due to the presence of local minima. Moreover, executing applications at a lower frequency results in a lower acceptance of applications since less CPUs are available. Thus, it is important to exploit such characteristics when designing the scheduling policy within a data center.

(a) Carbon emission. (b) Profit.

Fig. 6. Comparison of lower bound and upper bound.

6.1.3. Comparison of lower bound and upper bound

To evaluate the performance of our algorithms in terms of carbon emission reduced and profit gained by the Cloud provider, we compare our algorithms with the theoretically unreachable bound.

Fig. 6 shows how different policies closely perform to the lower bound of average carbon emission and the upper bound of average profit. In Fig. 6(a), the difference in average carbon emission for carbon emission based policies (GMCE, MCE–MCE, and MCE–MP) and the lower bound is less than about 16% which becomes less than about 2% in the case of 20% HU applications. On the other hand, in Fig. 6(b), the difference in average profit for profit based policies (GMP and MP–MP) and the upper bound is less than about 2% which becomes less than about 1% in the case of 40% of HU applications. Hence, in summary, our carbon emission based and profit based policies perform within about 16% and 2% of the optimal carbon emission and profit respectively.

In Fig. 6(a) and (b), with the increase in HU applications, the difference between the lower/upper bounds and various policies is increasing. This is due to the increase in looseness of the bounds with the increase in HU applications. To avoid deadline misses with a higher number of HU applications, our proposed policies schedule more applications at higher CPU frequency which results in higher energy consumption. This in turn leads to an increase in the carbon emission and decrease in the profit. Whereas, for computing the lower/upper bounds, we only consider energy consumption at the optimal CPU frequency. Thus, the effect of urgency on the bounds is not as considerable as in our policies. This explains why our policies are closer to the bounds for a lower number of HU applications.

6.1.4. Impact of urgency and arrival rate of applications

Fig. 7 shows how the urgency and arrival rate of applications affects the performance of carbon emission based policies (GMCE, MCE–MCE, and MCE–MP) and profit based policies (GMP and MP–MP). The metrics of total carbon emission and total profit are used since the Cloud provider needs to know the collective loss in carbon emission and gain in profit across all data centers.

When the number of HU applications increases, the total profit of all policies (Fig. 7(c)) decreases almost linearly by about 45% from 0% to 100% HU applications. Similarly, there is also a drop in total carbon emission (Fig. 7(a)). This fall in total carbon emission and total profit is due to the lower acceptance of applications as observed in Fig. 7(e). In Fig. 7(a), the decrease in total carbon emission for profit based policies (GMP and MP–MP) is much more than that of carbon emission based policies (MCE–MP, GMCE, and MCE–MCE). This is because carbon emission based policies schedule applications on more carbon-efficient data centers.

Likewise, the increase in arrival rate also affects the total carbon emission (Fig. 7(b)) and total profit (Fig. 7(d)). As more applications are submitted, less applications can be accepted (Fig. 7(f)) since it is harder to satisfy their deadline requirement when workload is high.

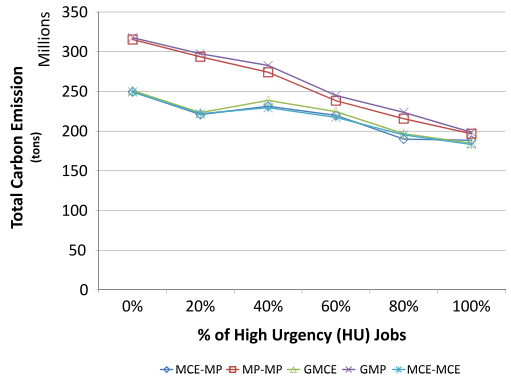
6.1.5. Impact of carbon emission rate

To examine the impact of carbon emission rate in different locations on our policies, we vary the carbon emission rate, while keeping all other factors such as electricity price as the same. Using normal distribution with *mean* = 0.2, random values are generated for the following three classes of carbon emission rate across all data centers as: (A) Low variation (low) with *standard deviation* = 0.05, (B) Medium variation (medium) with *standard deviation* = 0.2, and (C) High variation (high) with *standard deviation* = 0.4. All experiments are conducted at medium job arrival rate with 40% of HU applications.

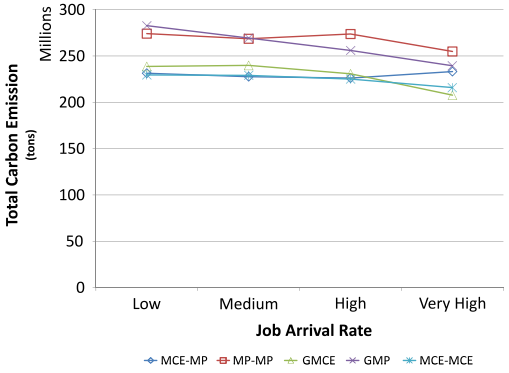
The performance of all policies is similar for all three cases of carbon emission rate. For example, in Fig. 8(a), the carbon emission of profit based policies (GMP and MP–MP) is always higher than carbon emission based policies (GMCE, MCE–MCE, and MCE–MP). Similarly, for profit (Fig. 8(b)), all profit based policies perform better than all carbon emission based policies. For instance, in Fig. 8(a), the difference in carbon emission of MCE–MCE and MP–MP is about 12% for low variation, which increases to 33% for high variation. On the other hand, in Fig. 8(b), the corresponding decrease in profit is almost negligible and is less than 1% for both the low and high variation case. Moreover, by comparing MCE–MCE and MP–MP in Fig. 8(c), the amount of workload executed by MCE–MCE is slightly higher than MP–MP. Thus, for the case of high variation in carbon emission rate, Cloud providers can use carbon emission based policies such as MCE–MCE to considerably reduce carbon emission with almost negligible impact on their profit. For minimizing carbon emission, MCE–MCE is preferred over GMCE since the latter leads to lower profit due to the scheduling of more applications on data centers with higher electricity price.

6.1.6. Impact of electricity price

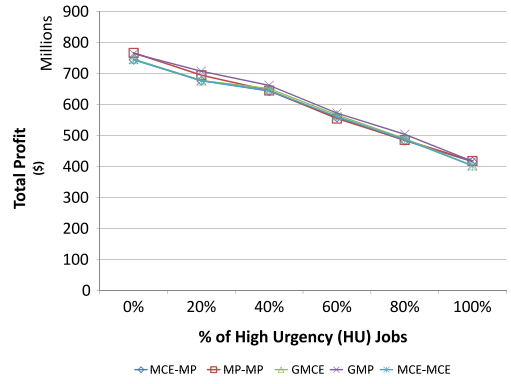
To investigate the impact of electricity price in different locations on our policies, we vary the electricity price, while keeping all other factors such as carbon emission rate as the same. Using normal distribution with *mean* = 0.1, random values are generated for the following three classes of electricity price across all data centers as: (A) Low variation (low) with *standard deviation* = 0.01, (B) Medium variation (medium) with *standard deviation* = 0.02, and (C) High variation (high) with *standard deviation* = 0.05. All experiments are conducted at medium job arrival rate with 40% of HU applications.



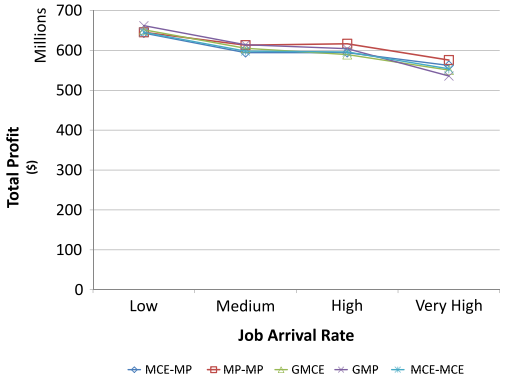
(a) Carbon emission vs. urgency.



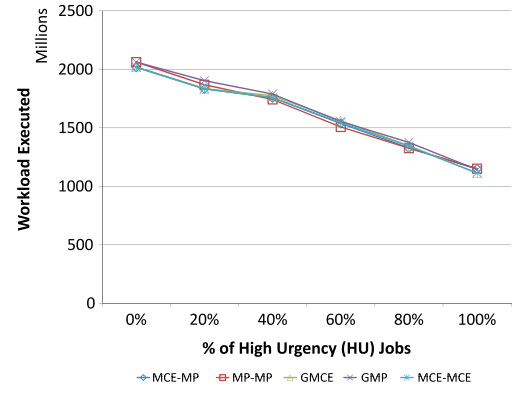
(b) Carbon emission vs. arrival rate.



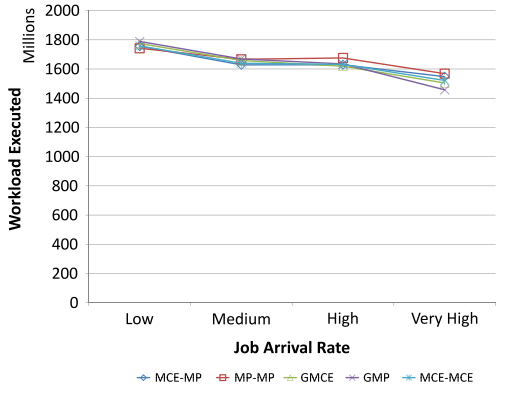
(c) Profit vs. urgency.



(d) Profit vs. arrival rate.



(e) Workload executed vs. urgency.



(f) Workload executed vs. arrival rate.

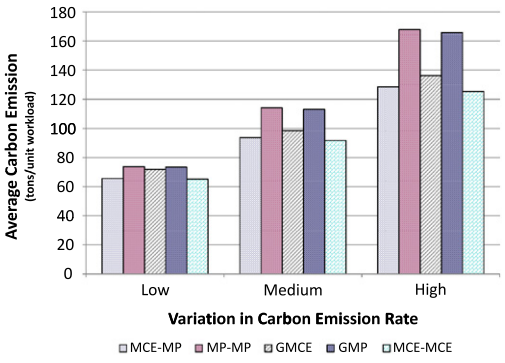
Fig. 7. Impact of urgency and arrival rate of applications.

The variation in electricity price affects the performance of profit based policies (GMP and MP–MP) in terms of carbon emission (Fig. 9(a)) and workload executed (Fig. 9(c)), while carbon emission based policies (GMCE, MCE–MCE and MCE–MP) are not affected. But, the profit of all policies decrease more as the variation of electricity price increases (Fig. 9(b)) due to the subtraction of energy cost from profit. For high variation in electricity price, there is not much difference (about 1.4%) in carbon emission between MP–MP and MCE–MCE (Fig. 9(a)). Hence, Cloud providers can use MP–MP which gives slightly better average profit than carbon emission based policies (GMCE, MCE–MCE and MCE–MP). On the other hand, for cases when the variation in electricity price is not high, providers can use carbon emission based policies such as MCE–MCE and MCE–MP to reduce about 5%–7% of carbon emission by sacrificing less than 0.5% of profit.

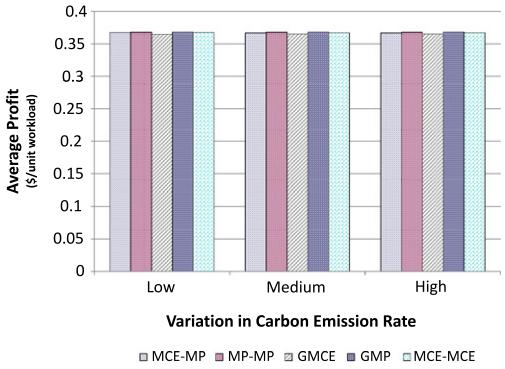
6.1.7. *Impact of data center efficiency*

To study the impact of data center efficiency in different locations on our policies, we vary the datacenterefficiency $= \frac{COP}{COP+1}$, while keeping all other factors such as carbon emission rate as the same. Using normal distribution with *mean* $= 0.4$, random values are generated for the following three classes of data center efficiency across all data centers as: (A) Low variation (low) with *standard deviation* $= 0.05$, (B) Medium variation (medium) with *standard deviation* $= 0.12$, and (C) High variation (high) with *standard deviation* $= 0.2$. All experiments are conducted at medium job arrival rate with 40% of HU applications.

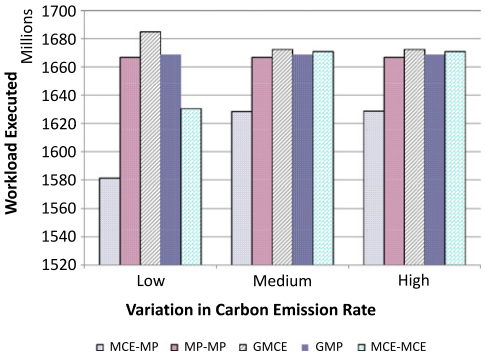
Fig. 10(a) shows carbon emission based policies (GMCE, MCE–MCE and MCE–MP) achieve the lowest carbon emission with almost equal values. MCE–MCE performs better than MCE–MP by scheduling more HPC workload (Fig. 10(c)) while achieving similar



(a) Carbon emission.

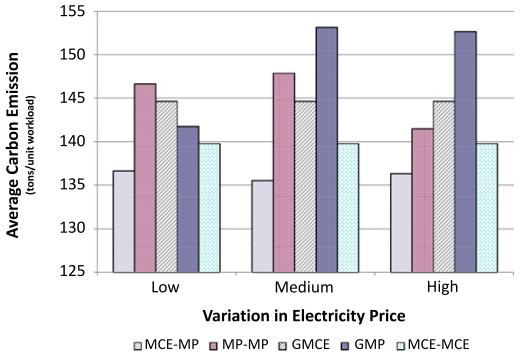


(b) Profit.

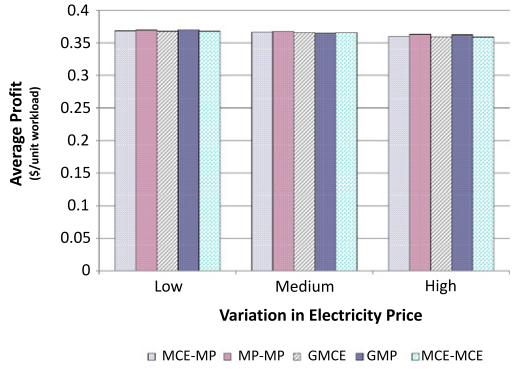


(c) Workload executed.

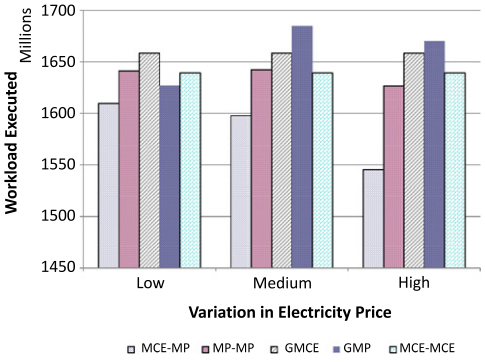
Fig. 8. Impact of carbon emission rate.



(a) Carbon emission.

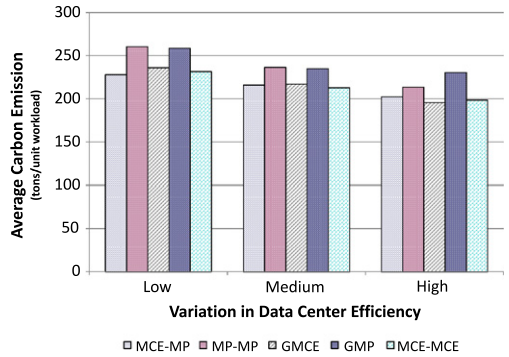


(b) Profit.

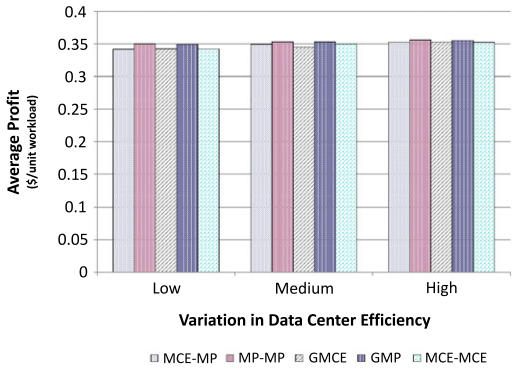


(c) Workload executed.

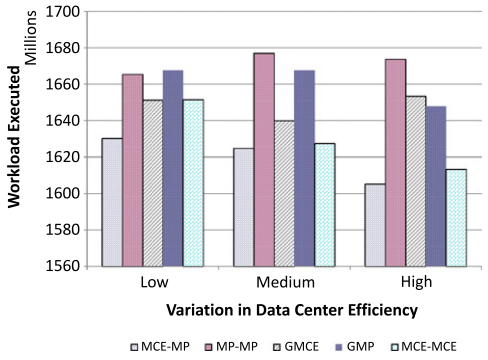
Fig. 9. Impact of electricity price.



(a) Carbon emission.



(b) Profit.



(c) Workload executed.

Fig. 10. Impact of data center efficiency.

profit (Fig. 10(b)). But when the variation in data center efficiency is high, GMCE can execute much higher workload (Fig. 10(c)) than MCE–MCE and MCE–MP while achieving only slightly less profit than profit based policies (GMP and MP–MP) (Fig. 10(b)). Thus, Cloud providers can use GMCE to decrease the carbon emissions across their data centers without significant profit loss.

6.2. Evaluation with data transfer cost

6.2.1. Impact of data transfer cost

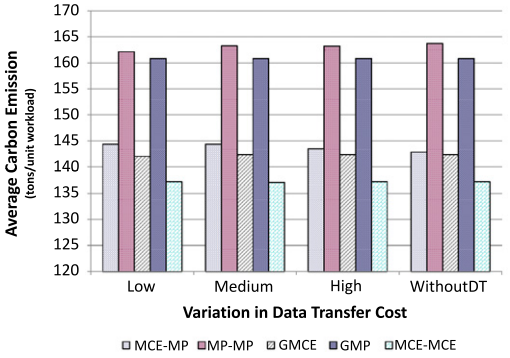
The data transfer cost of the Cloud provider varies across different data centers. Thus, to study the impact of data transfer cost on our policies, we vary the data transfer cost while keeping all other factors such as carbon emission rate and electricity price as the same. Since this paper focuses on compute-intensive parallel applications with low data transfer requirements, the maximum data transfer size of an application is set to only 10 TB. For this set of experiments, the Cloud provider charges the user a fixed price of $0.17 /GB for data transfer up to 10 TB, which is derived from Amazon EC2 [1]. Since, the workload traces used for the experiments does not contain any information on input or output data, thus the data transferred during execution is randomly generated during simulation. The data transfer size of an application is varied between [0, 10] TB using uniform distribution. The data transfer cost that the Cloud provider has to incur is varied between $[0, 0.17] using normal distribution with $mean = 0.4 * 0.17$. Random values are generated for the following three classes of data transfer cost across all data centers as: (A) Low variation (low) with *standard deviation* $= 0.05$, (B) Medium variation (medium) with *standard deviation* $= 0.12$, and (C) High variation (high) with *standard deviation* $= 0.2$. All experiments are conducted at medium job arrival rate with 40% of HU applications.

Fig. 11 shows how the average carbon emission and profit will be affected due to data transfer cost in comparison to the case when data transfer cost is not considered (as indicated by WithoutDT). The relative performance of all policies has remained almost the same even with data transfer cost. For instance, in Fig. 11(a) and (c), MP–MP results in the maximum average carbon emission, while MCE–MCE results in the minimum carbon emission. This is because of the compute-intensive workload, whereby the impact of data transfer cost is negligible in comparison to the execution cost. There is only a slight increase in the average profit (Fig. 11(b)) due to the additional profit gained by the Cloud provider from the transfer of data.

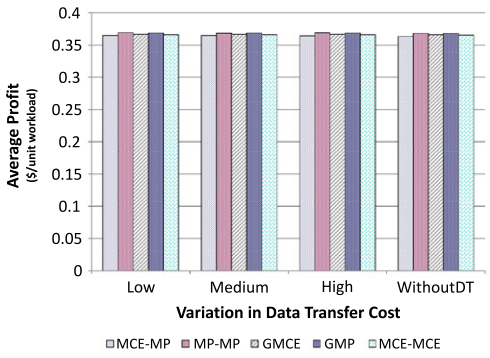
7. Concluding remarks and future directions

The usage of energy has become a major concern since the price of electricity has increased dramatically. Especially, Cloud providers need a high amount of electricity to run and maintain their computational resources in order to provide the best service level for the customer. Although this importance has been emphasized in a lot of research literature, the combined approach of analyzing the profit and energy sustainability in the resource allocation process has not been taken into consideration.

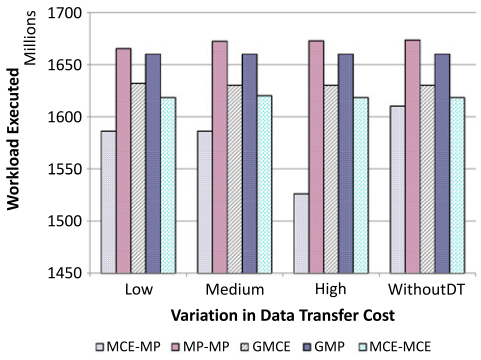
The goal of this paper is to outline how managing resource allocation across multiple locations can have an impact on the energy cost of a provider. The overall meta-scheduling problem is described as an optimization problem with dual objective functions. Due to its NP-hard characteristic, several heuristic policies are proposed and compared. The policies are compared with each other for different scenarios and also with the derived lower/upper bounds. In some cases, the policies performed very well with only almost 1% away from the upper bound of profit. By introducing DVS and hence lowering the supply voltage of CPUs, the energy cost for executing HPC workloads can be reduced by 33% on average. Applications will run on CPUs with a lower frequency than expected, but they still meet the required deadlines. The limitation



(a) Carbon emission.



(b) Profit.



(c) Workload executed.

Fig. 11. Impact of data transfer cost.

Table 4
Summary of heuristics with comparison results.

Meta-scheduling policy	Description	Time Complexity	Overall performance				
			HU Jobs	Arrival rate	Carbon emission rate	Data center efficiency	Energy cost
GMCE	Greedy (Carbon Emission)	$O(NJ)$	Bad	Bad	Bad	Best (high)	Bad
MCE–MCE	Two-phase Greedy (Carbon Emission)	$O(NJ^2)$	Good (low)	Good (low)	Best (high)	Okay (low)	Good (low)
GMP	Greedy (Profit)	$O(NJ)$	Okay (high)	Okay (high)	Bad (low)	Bad (high)	Bad
MP–MP	Two-phase Greedy (Profit)	$O(NJ^2)$	Good (high)	Bad (Carbon Emission), Best (Profit)	Good (low)	Best (low)	Good (high)
MCE–MP	Two-phase Greedy (Carbon Emission and Profit)	$O(NJ^2)$	Best (low)	Good (high)	Okay	Okay	Best (low)

of carbon emission can be forced by governments to comply with certain threshold values [15]. In such cases, Cloud providers can focus on reducing carbon emission in addition to minimizing energy consumption.

We identified that policies like MCE–MCE can help provider to reduce their emission while almost maintaining their profit. If the provider faces a volatile electricity price, the MP–MP policy will lead to a better outcome. Depending on the environmental and economic constraints, Cloud providers can selectively choose different policies to efficiently allocate their resources to meet customers' requests. The characteristics and performance of each meta-scheduling policy are summarized in Table 4, where "low" and "high" represent the scenario for which the overall performance of the policy is given. For instance, GMCE performs the best when the variation in data center efficiency is high, while MCE–MP performs the best when the variation in energy cost is low or when there is a low number of HU applications. We observed that the impact of data transfer cost is minimal for the compute-intensive applications that we have considered. However, our model has explicitly considered the data transfer cost and thus can be used for data-intensive applications as well.

In future, we will like to extend our model to consider the aspect of turning servers on and off, which can further reduce energy consumption. This requires a more technical analysis of the delay and power consumption for suspending servers, as well as the effect on the reliability of computing devices. We will also want to extend our policies for virtualized environments, where it can be easier to consolidate many applications on fewer physical servers. In addition, we can consider the energy (and potentially latency) overhead of moving data sets between the data centers, in particular for data-intensive HPC applications. This overhead can be quite significant depending on the size of the data set and the activity of the workloads.

Acknowledgments

We would like to thank Marcos Dias de Assuncao for his constructive comments on this paper. This work is partially supported by research grants from the Australian Research Council (ARC) and Australian Department of Innovation, Industry, Science and Research (DIISR).

Appendix A. Proof of 2-dimensional bin-packing problem

Definition 1. Let $L = (x_1, \ldots, x_{ji}, \ldots, x_n)$ be a given list of n items with a value of $x_{ji} \in (0, 1]$, and $B = b_1, \ldots, b_m$ be a finite sequence of m bins each of unit capacity. The 2-dimensional bin-packing problem is to assign each x_{ji} into a unique bin, with the sum of numbers in each $b_j \in B$ not exceeding one, such that the total number of used bins is a minimum (denoted by $L*$) [31].

Proposition 1. *The optimization problem described in Eqs. (10) and (11) is an NP-hard problem.*

Proof. This proposition can be easily proven by reducing the problem to the (2-dimensional) bin-packing problem [31], which is a well-known NP-hard problem. The number of bins m is equal to the available N data centers. The dimensions of an application j consist of two parameters d_j and e_{ji}. However, e_{ji} depends on the frequency of the CPUs of data center i. By defining a transformation function $\rho : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$, we can transform e_{ji} to e_j. This restriction only considers data centers with the same frequency for all CPUs. Consequently, $f(d_j, e_j) = x_{ji}$ and by Definition 1, it is a 2-dimensional bin-packing problem defined by the deadline and runtime of an application. □

Appendix B. Analysis of exploiting local minima in DVS

Section 3.4 shows that the energy consumption of a CPU depends on the frequency of the CPU at which an application will be executed. Hence the objective is to obtain an optimal CPU frequency so that the energy consumption of the CPU can be minimized while completing the application within its deadline. From the plot of energy consumption in Fig. B.1, we can observe the existence of the local minima where the energy consumption will be the minimum. In order to identify this local minima, we differentiate the energy consumption of an application j on a CPU at a data center i with respect to the operating CPU frequency f_{ij} as:

$$E_{ij}^c = (\beta_i + \alpha_i (f_{ij})^3) \times n_j e_{ji} \times \left(\gamma_j^{cpu} \left(\frac{f_i^{max}}{f_{ij}} - 1 \right) + 1 \right) \tag{B.1}$$

$$\frac{\partial (E_{ij}^c)}{\partial f_{ij}} = n_j e_{ji} \times \left[\frac{(\beta_i + \alpha_i (f_{ij})^3) f_i^{max} \gamma_j^{cpu}}{(f_{ij})^2} + 3\alpha_i (f_{ij})^2 \left(1 + \left(-1 + \frac{f_i^{max}}{f_{ij}} \right) \gamma_j^{cpu} \right) \right]. \tag{B.2}$$

For local minima,

$$\frac{\partial (E_{ij}^c)}{\partial f_{ij}} = 0 \tag{B.3}$$

$$\frac{\left(\beta_i + \alpha_i (f_{ij})^3\right) f_i^{max} \gamma_j^{cpu}}{(f_{ij})^2} + 3\alpha_i (f_{ij})^2 \left(1 + \left(-1 + \frac{f_i^{max}}{f_{ij}} \right) \gamma_j^{cpu} \right) = 0. \tag{B.4}$$

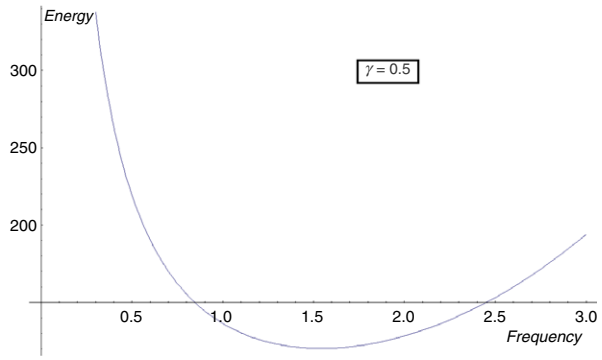


Fig. B.1. Energy consumption vs. CPU frequency.

Since we can clearly see that the local minima exists in Fig. B.1, at least one root of the above polynomial will exist in the range $[0, \infty]$. When $\gamma_j^{cpu} = 1$, the above equation will reduce to:

$$\frac{-\beta_i}{(f_{ij})^2} + 2\alpha_i f_{ij} = 0. \tag{B.5}$$

Many previous work [20,28] have chosen a fixed γ_j^{cpu} value to compute the energy usage of CPUs. Likewise, in this paper, we assume a fixed $\gamma_j^{cpu} = 1$ to understand the worst case scenario of CPU energy usage. We can then pre-compute the local minima (with static variables such as CPU power efficiency) before starting the meta-scheduling algorithm.

References

[1] Amazon, Amazon Elastic Compute Cloud (EC2), Aug. 2009. http://www.amazon.com/ec2/.
[2] Alpiron, Alpiron Suite, 2009. http://www.alpiron.com.
[3] C. Belady, In the data center, power and cooling costs more than the it equipment it supports, Electronics Cooling 13 (1) (2007) 24.
[4] F. Berman, H. Casanova, A. Chien, K. Cooper, H. Dail, A. Dasgupta, W. Deng, J. Dongarra, L. Johnsson, K. Kennedy, C. Koelbel, B. Liu, X. Liu, A. Mandal, G. Marin, M. Mazina, J. Mellor-Crummey, C. Mendes, A. Olugbile, J.M. Patel, D. Reed, Z. Shi, O. Sievert, H. Xia, A. YarKhan, New grid scheduling and rescheduling methods in the grads project, International Journal of Parallel Programming 33 (2) (2005) 209–229.
[5] R. Bianchini, R. Rajamony, Power and energy management for server systems, Computer 37 (11) (2004) 68–74.
[6] D. Bradley, R. Harper, S. Hunter, Workload-based power management for parallel computer systems, IBM Journal of Research and Development 47 (5) (2003) 703–718.
[7] T. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther, J. Robertson, M. Theys, B. Yao, D. Hensgen, et al., A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems, Journal of Parallel and Distributed Computing 61 (6) (2001) 810–837.
[8] T. Burd, R. Brodersen, Energy efficient CMOS microprocessor design, in: Proceedings of the 28th Hawaii International Conference on System Sciences, HICSS'95, vol. 1060, 1995.
[9] J. Burge, P. Ranganathan, J.L. Wiener, Cost-aware scheduling for heterogeneous enterprise machines (CASH'EM), Technical Report HPL-2007-63, HP Labs, Palo Alto, Apr. 2007.
[10] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems 25 (6) (2009) 599–616.
[11] J.S. Chase, D.C. Anderson, P.N. Thakar, A.M. Vahdat, R.P. Doyle, Managing energy and server resources in hosting centers, SIGOPS Operating Systems Review 35 (5) (2001) 103–116.
[12] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, N. Gautam, Managing server energy and operational costs in hosting centers, ACM SIGMETRICS Performance Evaluation Review 33 (1) (2005) 303–314.
[13] M. Chin, Desktop cpu power survey, Silentpcreview. com, 2006.
[14] 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.
[15] K. Corrigan, A. Shah, C. Patel, Estimating environmental costs, in: Proceedings of the Ist USENIX Workshop on Sustainable Information Technology, San Jose, CA, USA, 2009.

[16] US Department of Energy, US Energy Information Administration (EIA) report, 2007. http://www.eia.doe.gov/cneaf/electricity/epm/table5_6_a.html.
[17] A. Elyada, R. Ginosar, U. Weiser, Low-complexity policies for energy-performance tradeoff in chip-multi-processors, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 16 (9) (2008) 1243–1248.
[18] United States Environmental Protection Agency, Letter to enterprise server manufacturer or other interested stakeholder, Dec. 2006. http://www.energystar.gov/ia/products/downloads/Server_Announcement.pdf.
[19] United States Environmental Protection Agency, Report to congress on server and data center energy efficiency, Public Law 109-431, Aug. 2007. http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf.
[20] M. Etinski, J. Corbalan, J. Labarta, M. Valero, A. Veidenbaum, Power-aware load balancing of large scale MPI applications, in: Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing, Rome, Italy, 2009.
[21] EUbusiness, Proposed EU regulation to reduce CO_2 emissions from cars, Dec. 2007. http://www.eubusiness.com/Environ/co2-cars-eu-guide/.
[22] X. Fan, W.-D. Weber, L.A. Barroso, Power provisioning for a warehouse-sized computer, in: Proceedings of the 34th Annual International Symposium on Computer Architecture, ACM, New York, NY, USA, 2007, pp. 13–23.
[23] D. Feitelson, Parallel workloads archive, Aug. 2009. http://www.cs.huji.ac.il/labs/parallel/workload.
[24] D.G. Feitelson, L. Rudolph, U. Schwiegelshohn, K.C. Sevcik, P. Wong, Theory and practice in parallel job scheduling, in: Proceedings of the 1997 International Workshop on Job Scheduling Strategies for Parallel Processing, London, UK, 1997.
[25] W. Feng, K. Cameron, The green500 list: encouraging sustainable supercomputing, Computer (2007) 50–55.
[26] X. Feng, R. Ge, K.W. Cameron, Power and energy profiling of scientific applications on distributed systems, in: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, Los Alamitos, CA, USA, 2005.
[27] W. Feng, T. Scogland, The green500 list: year one, in: Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing, Rome, Italy, 2009.
[28] V. Freeh, D. Lowenthal, F. Pan, N. Kappiah, R. Springer, B. Rountree, M. Femal, Analyzing the energy-time trade-off in high-performance computing applications, IEEE Transactions on Parallel and Distributed Systems 18 (6) (2007) 835.
[29] A. Gandhi, M. Harchol-Balter, R. Das, C. Lefurgy, Optimal power allocation in server farms, in: Proceedings of the 11th International Joint Conference on Measurement and Modeling of Computer Systems, Seattle, WA, USA, 2009.
[30] 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.
[31] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, San Francisco, USA, 1979.
[32] S. Greenberg, E. Mills, B. Tschudi, P. Rumsey, B. Myatt, Best practices for data centers: results from benchmarking 22 data centers, in: Proceedings of the 2006 ACEEE Summer Study on Energy Efficiency in Buildings, Pacific Grove, USA, 2006.
[33] T. Hardware, Cpu performance charts, Toms Hardware, 2006.
[34] C. Hsu, U. Kremer, The design, implementation, and evaluation of a compiler algorithm for CPU energy reduction, in: Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation, Sweden, 2003.
[35] O. Ibarra, C. Kim, Heuristic Algorithms for Scheduling Independent Tasks on Nonidentical Processors, Journal of the ACM 24 (2) (1977) 280–289.
[36] 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.
[37] S.-H. Jang, V.E. Taylor, X. Wu, M. Prajugo, E. Deelman, G. Mehta, K. Vahi, Performance prediction-based versus load-based site selection: quantifying the difference, in: M. J. Oudshoorn, S. Rajasekaran (Eds.), ISCA PDCS, ISCA, 2005, pp. 148–153.
[38] K. Kim, R. Buyya, J. Kim, Power aware scheduling of bag-of-tasks applications with deadline constraints on dvs-enabled clusters, in: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid, Rio de Janeiro, Brazil, 2007.
[39] B. Lawson, E. Smirni, Power-aware resource allocation in high-end systems via online simulation, in: Proceedings of the 19th Annual International Conference on Supercomputing, Cambridge, USA, 2005.
[40] J. Markoff, S. Hansell, Hiding in Plain Sight, google seeks more power, 2006. http://www.nytimes.com/2006/06/14/technology/14/search.html.
[41] S. Martello, P. Toth, An algorithm for the generalized assignment problem, Operational Research 81 (1981) 589–603.
[42] J. Moore, J. Chase, P. Ranganathan, R. Sharma, 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.
[43] A.W. Mu'alem, D.G. Feitelson, Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling, IEEE Transactions on Parallel and Distributed Systems 12 (6) (2001) 529–543.
[44] G.R. Nudd, D.J. Kerbyson, E. Papaefstathiou, S.C. Perry, J.S. Harper, D.V. Wilcox, Pace–a toolset for the performance prediction of parallel and distributed systems, International Journal of High Performance Computing Application 14 (3) (2000) 228–251.
[45] A. Orgerie, L. Lefèvre, J. Gelas, 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.
[46] C. Patel, R. Sharma, C. Bash, M. Beitelmal, Energy flow in the information technology stack: coefficient of performance of the ensemble and its impact on the total cost of ownership, HP Labs External Technical Report, HPL-2006-55.
[47] C. Patel, R. Sharma, C. Bash, S. Graupner, Energy aware grid: global workload placement based on energy efficiency, Technical Report HPL-2002-329, HP Labs, Palo Alto, Nov. 2002.
[48] P. Pillai, K. Shin, Real-time dynamic voltage scaling for low-power embedded operating systems, in: Proceedings of the 18th ACM Symposium on Operating Systems Principles, Banff, Canada, 2001.
[49] R. Porter, Mechanism design for online real-time scheduling, in: Proceeding of the 5th ACM Conference on Electronic Commerce, New York, USA, 2004.
[50] S. Rivoire, M.A. Shah, P. Ranganathan, C. Kozyrakis, Joulesort: a balanced energy-efficiency benchmark, in: SIGMOD'07: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, 2007, pp. 365–376.
[51] C. Rusu, A. Ferreira, C. Scordino, A. Watson, Energy-efficient real-time heterogeneous server clusters, in: Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium, Stockholm, Sweden, 2006.
[52] V. Salapura, et al. Power and performance optimization at the system level, in: Proceedings of the 2nd Conference on Computing Frontiers, Ischia, Italy, 2005.
[53] H.A. Sanjay, S. Vadhiyar, Performance modeling of parallel applications for grid scheduling, Journal of Parallel Distributed Computing 68 (8) (2008) 1135–1145.
[54] G. Singh, C. Kesselman, E. Deelman, A provisioning model and its comparison with best-effort for performance-cost optimization in grids, in: Proceedings of the 16th International Symposium on High Performance Distributed Computing, California, USA, 2007.
[55] W. Smith, I. Foster, V. Taylor, Predicting application run times using historical information, in: D.G. Feitelson, L. Rudolph (Eds.), Job Scheduling Strategies for Parallel Processing, in: Lect. Notes Comput. Sci., vol. 1459, Springer Verlag, 1998, pp. 122–142.
[56] Q. Tang, S.K.S. Gupta, D. Stanzione, P. Cayton, 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.
[57] G. Tesauro, et al. Managing power consumption and performance of computing systems using reinforcement learning, in: Proceedings of the 21st Annual Conference on Neural Information Processing Systems, Vancouver, Canada, 2007.
[58] TOP500 Supercomputers, Supercomputer's Application Area Share, 2009. http://www.top500.org/stats/list/33/apparea.
[59] G. Verdun, D. Azevedo, H. Barrass, S. Berard, M. Bramfitt, T. Cader, T. Darby, C. Long, N. Gruendler, B. Macarthur, et al. The green grid metrics: data center infrastructure efficiency (DCIE) detailed analysis, the green grid.
[60] L. Wang, Y. Lu, Efficient power management of heterogeneous soft real-time clusters, in: Proceedings of the 2008 Real-Time Systems Symposium, Barcelona, Spain, 2008.

Saurabh Kumar Garg is a Ph.D. student at the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne, Australia. In Melbourne University, he has been awarded various special scholarships for his Ph.D. candidature. He completed his 5-year Integrated Master of Technology in Mathematics and Computing from the Indian Institute of Technology (IIT) Delhi, India, in 2006. His research interests include Resource Management, Scheduling, Utility and Grid Computing, Cloud Computing, Green Computing, Wireless Networks, and Ad hoc Networks.

Chee Shin Yeo is a research engineer at the Institute of High Performance Computing (IHPC), Singapore. He completed his Ph.D. at the University of Melbourne, Australia. His research interests include parallel and distributed computing, services and utility computing, energy-efficient computing, and market based resource allocation.

Arun Anandasivam is a research assistant and Ph.D. student in the Institute of Information Systems and Management at Universität Karlsruhe. His research work comprises pricing policies and decision frameworks of grid and Cloud computing providers.

Rajkumar Buyya is a Professor of Computer Science and Software Engineering; and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft Pty Ltd., a spin-off company of the University, commercialising innovations originating from the CLOUDS Lab. He has pioneered Economic Paradigm for Service-Oriented Grid computing and demonstrated its utility through his contributions to conceptualisation, design and development of Cloud and Grid technologies such as Aneka, Alchemi, Nimrod-G and Gridbus that power the emerging eScience and eBusiness applications.