A Framework and Task Allocation Analysis for Infrastructure Independent Energy-Efficient Scheduling in Cloud Data Centers

B. Primas*, P. Garraghan†, D. W. McKee*, J. Summers‡, J. Xu*

School of Computing*
University of Leeds
Leeds, UK
{ scbjp, d.w.mckee, j.xu }@leeds.ac.uk

School of Computing and Communications†
Lancaster University
Lancaster, UK
p.garraghan@lancaster.ac.uk

School of Mechanical Engineering‡
University of Leeds
Leeds, UK
j.l.summers@leeds.ac.uk

Abstract*—Cloud computing represents a paradigm shift in provisioning on-demand computational resources underpinned by data center infrastructure, which now constitutes 1.5% of worldwide energy consumption. Such consumption is not merely limited to operating IT devices, but encompasses cooling systems representing 40% total data center energy usage. Given the substantive complexity and heterogeneity of data center operation spanning both computing and cooling components, obtaining analytical models for optimizing data center energy-efficiency is an inherently difficult challenge. Specifically, difficulties arise pertaining to the non-intuitive relationship between computing and cooling energy in the data center, computationally complex energy modeling, as well as cooling models restricted to a specific class of data center facility geometry - all of which arise from the interdisciplinary nature of this research domain. In this paper we propose a framework for energy-efficient scheduling to alleviate these challenges. It is applicable to any type of data center infrastructure and does not require complex modeling of energy. Instead, the concept of a target workload distribution is proposed. If the workload is assigned to nodes according to the target workload distribution, then the energy consumption is minimized. The exact target workload distribution is unknown, but an approximated distribution is delivered by the framework. The scheduling objective is to assign workload to nodes such that the workload distribution becomes as similar as possible to the target distribution in order to reduce energy consumption. Several mathematically sound algorithms have been designed to address this novel type of scheduling problem. Simulation results demonstrate that our algorithms reduce the relative deviation by at least 16.9% and the relative variance by at least 22.67% in comparison to (asymmetric) load balancing algorithms.

***Keywords*—Cloud Computing; Energy Efficiency; Workload Scheduling; Thermal-Aware Scheduling; Scheduling Heuristics; Combinatorial Optimization**

I. Introduction

Cloud computing has established itself as a fundamental aspect within modern internet infrastructure. As such, Cloud computing is used by providers to deliver IT services as a utility: customers have access to on-demand service enforced by a Service Level Agreement (SLA). In order to stay competitive, Cloud providers face numerous challenges towards efficient operation with energy-efficiency emerging as a key requirement. This is made apparent when considering that in 2010 world-wide energy consumption of data centers - the physical infrastructure of Cloud computing - accounted for up to 1.5 % of the world-wide total energy usage [1], while data centers in the US accounted for up to 2.2 % of the total energy consumption of the US [2].

Job scheduling is a core component in Cloud computing systems as it significantly impacts numerous aspects of provisioned service performance, as well as energy reduction of a system through node load balancing, workload consolidation, and temperature-aware scheduling. Although intensively studied, energy-aware scheduling within data centers still faces numerous challenges that remain unsolved. First, data center energy is composed by both computing and cooling components, and the interaction between these components is highly complex in nature. For example, an intuitive approach to reduce energy consumption is workload consolidation - migrate workload from low utilization nodes such that nodes can be switched off, thus reducing computing energy. However, this approach leads to the formation of hot spots within data centers resulting in increased cooling energy [3], [4]. This trade-off is inherently difficult to understand and leads to sub-optimal reduction in data center energy [5]. Second, analytical modeling of energy consumption within the data center is a substantive task, especially when modeling cooling energy which captures various fluid mechanical and thermodynamic aspects of the system. Lastly, existing approaches that directly model total cooling energy are typically restricted to a certain class of data center architecture [5], [6], [7]. In particular, cooling models heavily depend on the data center geometry (i.e. facility layout) and location of installed cooling devices such as Computer Room Air Conditioners (CRACs).

In this paper, we propose a framework that allows to conduct effective energy-aware scheduling without dependencies pertaining to assumptions of the underlying data center infrastructure. The framework does not need to make use of analytical models of energy consumption. Instead, the problem of energy-efficient scheduling is divided into components that can be studied and applied independently from each other. Each component makes either use of data center thermal management, machine learning or mathematical optimization theory. Such an approach abstracts away infrastructure dependency requirements using the concept of a *target utilization distribution*. For an arbitrary but fixed type of data center (i.e. a specific size, geometry, cooling system, etc.), a recommendation pertaining to a target workload distribution that is expected to be energy-efficient is given by a system or expert. The scheduler then assigns tasks to nodes so that the actual workload distribution is as "similar" as possible to the target workload distribution. The expert or system is expected to determine the target workload distribution based on calculations and does not have the opportunity to execute workload. Therefore, the target workload distribution may have room for further improvement with respect to energy-efficiency. To this end, a machine learning component is embedded within the framework in order to improve the target workload distribution over time.

This paper focuses on the task allocation component of the framework, i.e. the challenge of assigning the workload to nodes such that the actual workload is distributed as "similar" as possible to the target workload distribution. In particular, we discover that the underlying scheduling model corresponds directly to the Generalized Assignment Problem (GAP). As such, there is a rich literature on theoretical and applied aspects of which we make extensively use of in our approach to enhance energy-aware scheduling. We propose scheduling algorithms and evaluate them against (asymmetric) load balancing algorithms by conducting comprehensive simulation of Bag-of-Task submissions to a data center. Our main contributions are listed as follows:

An infrastructure independent framework for energy-efficient scheduling in Cloud data centers. The framework does not make any assumptions on the data center infrastructure and thus, in principle, can be applied to any form of data center. There is no complex energy modeling required. Instead, energy-efficient scheduling decisions are made based on a target workload distribution.

Establish a link between energy-aware scheduling algorithms in Clouds and the GAP. We abstract away technical components of a data center which allows us to formulate the scheduling problem as an Integer Linear Programming problem. These structural insights enable us to establish a link to the GAP. Our proposed algorithms outperform traditional (asymmetric) load balancing algorithms.

The rest of the paper is organized as follows. Section II discusses related work. In Section III the framework of our approach is explained. The system and application model is introduced in Section IV. In Section V the algorithmic solutions to the scheduling problem are presented. A simulation-based evaluation is conducted in Section VI and Section VII contains conclusions and provides future work.

II. Related Work

The majority of energy-aware scheduling for data centers has addressed the challenge from two perspectives: reducing computing energy (required to operate IT devises) and reducing cooling energy (required to keep IT devices at an acceptable temperature). These challenges have been addressed both in isolation and combined using a variety of different techniques from different fields. This section gives an overview of some of the relevant work for each category.

Computing energy can be reduced through workload consolidation [8] - with energy reduction achieved by turning off idle nodes - and has been intensively studied, see [9], [10], [11]. This problem is typically modeled as a bin packing problem which is NP-hard. However, a large variety of heuristics have been proposed over the years, see surveys [12], [13]. Another technique is Dynamic Voltage and Frequency Scaling (DVFS) that allows for controlling the operating frequency and voltage to reduce computing energy. This technique has been investigated both by applied researchers and practitioners of Cloud computing [14], [15], [16] as well as by theoreticians from mathematical perspectives [17], [18].

Reducing cooling power has received increasing attention in recent years, especially within the Mechanical Engineering research community. Cooling energy constitutes approximately 40% of the total data center energy consumption, and is responsible for rejecting heat from the facility created from the operation of IT devices. However, cooling energy depends on various aspects including the installed cooling system type, data center infrastructure or the location and climate zone of the data center [19]. Computational Fluid Dynamics (CDF) is a powerful yet time consuming technique in order to construct temperature models of nodes within a data center. It also makes a number of restrictive assumptions such as air ducting designs for data centers [20], referred to as supply respectively return schema. CFD models have been intensively studied [5], [21], [22], [23] and have been found to be capable of significantly reducing cooling energy however at the cost of high complexity and long modeling time. For example, it has been demonstrated [5] that combining computing and cooling into a total energy model is an effective means to reduce energy consumption. However, in order to achieve that infrastructure depend assumptions have been made that restrict the model to data centers with a raised floor and ceiling return air ducting design schemata.

III. The Framework

The core concept of the framework is based on an observation presented in the following. By considering the workload distribution of thermal-aware scheduling algorithms at different times, it is apparent that workload is not uniformly distributed across nodes. This results from diversity in hardware heterogeneity, variance in task allocation and resource usage, as well as physical location within the facility. For example, nodes which are situated closer to a CRAC can be cooled more energy-efficiently than nodes located further away from cooling devices. As a consequence, nodes closer to CRACs may receive a higher workload due to the relatively small energy required for cooling. Figure 1 illustrates this phenomenon observed in [5] for a data center consisting of 24 nodes with 4 CRACs. Nodes located close to CRACs receive a higher workload (up to 9% of the data center workload), whereas nodes located further away receive a significantly lower workload. Note that a percent value of 9% does not signify that the corresponding node is utilized to 9%, but that 9% of the data center workload is assigned to the node.

This observation suggests that a key aspect of thermal-aware scheduling is determining an appropriate workload distribution pattern. In all previous works, this is typically achieved by modeling cooling energy analytically using sophisticated and complex techniques such as Computational Fluid Dynamics (CFD). In this paper, we propose a framework that does not require complex analytical models to find an appropriate

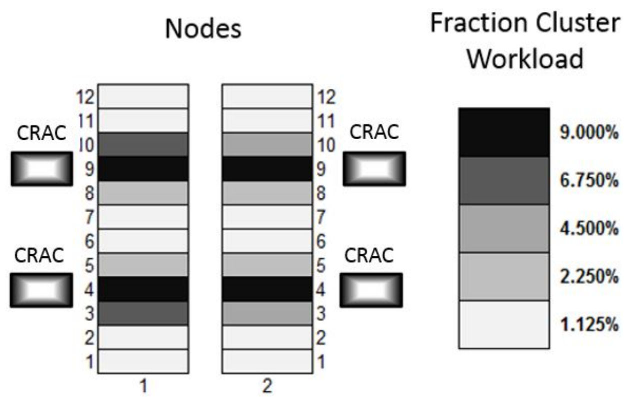

Fig. 1. Thermal-aware workload distribution for 24 nodes and 4 CRACs.

workload distribution. This is useful as it makes it much easier for developers to implement a proposed scheduling algorithm. Furthermore, our approach can be applied to any type of data center which is not the case for algorithms based on CFD. The framework is illustrated in Figure 2 and consists of the following components:

- Scientist: A domain-specific expert gives a recommendation of an appropriate workload distribution based on previous experience and supporting tools. This recommendation is used to initiate the machine learning procedure and is not required to be of high quality. Recommendations do not require actual execution of workload but are based on previous experience and supporting tools.
- Task Allocation: Responsible for assigning tasks to nodes such that the workload distribution becomes as similar as possible to the target workload distribution.
- Migration: Responsible for task migration in order to perform consolidation.
- Consolidation: Determines how many and which nodes should be turned off in order to save computing energy when data center utilization is low. When node M with target utilization fraction of ρ is turned off, the target utilization fraction of M is set to $\rho = 0$ and the target utilization fraction of other nodes are adjusted by factor $\frac{1}{1-p}$. Note that after this transformation the target utilization fractions still sum up to 1. As a consequence, the framework can be applied without any further adjustments when nodes are dynamically turned-on or off.
- Improvement: Enhances the initial workload distribution obtained by the scientist. Improvement of workload distribution is performed via machine learning techniques.

In this work, we focus on the task allocation component, i.e. developing scheduling algorithms that assign tasks to nodes such that the *actual workload* distribution approximates the target workload distribution as good as possible. Formally, for a data center consisting of m potentially heterogeneous nodes M_i with $1 \le i \le m$, we are given target utilization fractions $\rho_i \in [0,1]$ that satisfy $\sum_{i=1}^{m} \rho_i = 1$. These target utilization fractions should be such that if M_i is utilized with a fraction of ρ_i of the data center workload over time, the cooling energy is expected to be minimized. Note that the target utilization fraction values ρ_i depend on the data center architecture and are obtained by the scientist component.

A. Metrics For Analyzing Workload Distribution

The problem of distributing the workload such that the actual workload distribution gets as close as possible to the target workload distribution, can be considered as an asymmetric load balancing problem. In an asymmetric load balancing problem, each node is assigned a weight and the total workload is then distributed according to such weights. For example, for two nodes M_1 and M_2 with weights $\rho_1 = \frac{1}{3}$ and $\rho_2 = \frac{2}{3}$, node M_2 should ideally receive twice as much as M_1. For scheduling problems in this work, the weight for M_i is ρ_i.

In the literature, there have been proposed a number of metrics to evaluate asymmetric load balancing algorithms. In this work we consider the *total imbalance level* (IBL) and also propose two new metrics termed *total relative deviation* (RD) and *relative variance* (RVar). The IBL metric is based on absolute comparisons whereas the RD and the RVar are based on relative comparisons.

Let $U_i(t)$ denote the CPU utilization (in terms of work) of M_i at time t and let $\mathcal{U}(t) := \sum_{i=1}^{m} U_i(t)$ be the total data center CPU utilization at time t. We refer to the average value over time by adding an overline to the symbol. For example, we write $\overline{U_i}$ to denote the average CPU utilization over time, i.e. $\overline{U_i} := \frac{1}{t_2-t_1}\int_{t_1}^{t_2} U_i(\tau)\mathrm{d}\tau$ if the time period under consideration is given by interval $[t_1, t_2]$. The total imbalance level IBL is then defined by

$$\mathrm{IBL} := \frac{1}{m}\sum_{i=1}^{m}\left(\overline{U_i} - \rho_i\overline{\mathcal{U}}\right)^2.$$

As the IBL is quadratic, large deviations are penalized more severely. This is not the case for our proposed RD metric which is described in the following. First we consider

$$\mathrm{RD}(t) := \frac{1}{m}\sum_{i=1}^{m}\frac{|U_i(t) - \rho_i\mathcal{U}(t)|}{\rho_i\mathcal{U}(t)}$$

which corresponds to the relative deviation at time t. The relative deviation is then defined by $\mathrm{RD} := \overline{RD}$. This metric compares two distributions based on first-order comparisons. Therefore we also introduce the second-order metric

$$\mathrm{RVar} := \frac{1}{m}\sum_{i=1}^{m}\frac{\mathrm{Var}\left(U_i(t)\right)}{\mathrm{Var}\left(\rho_i\mathcal{U}(t)\right)}.$$

We believe that all three metrics together give sufficient information for an appropriate comparison.

IV. System and Application Model

The chosen system model is depicted in Figure 3 and based on the model proposed in [24]. Although we consider a similar system model, the research focus of [24] is to achieve energy-efficiency using DVFS techniques - which is substantially different to this work's research objectives (and as a result, would not represent a suitable comparison for evaluation).

We consider a virtualized IaaS Cloud data center composed of m potentially heterogeneous hosts M_i with $1 \le i \le m$. Node M_i consists of γ_i identical cores. We assume that each VM operates on one core and that VMs do not share CPU

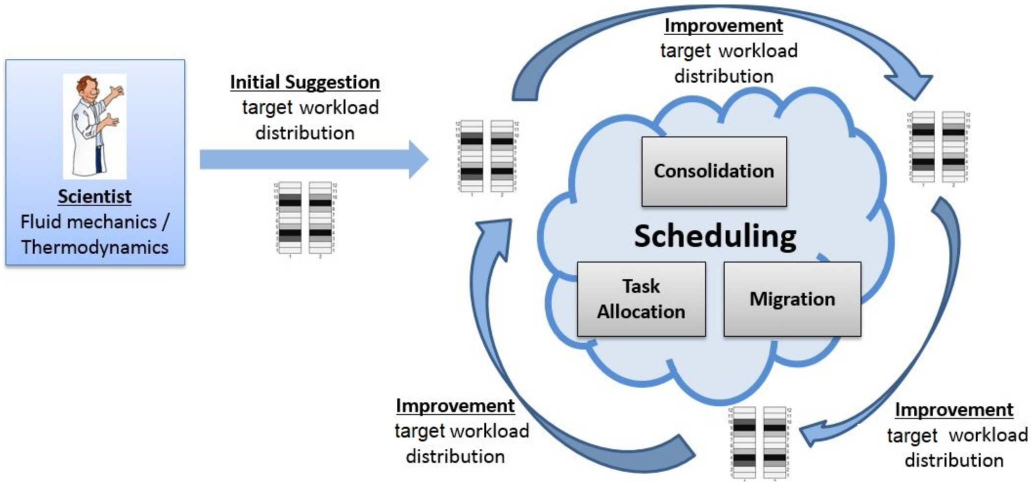
Scientist
Fluid mechanics / Thermodynamics
Initial Suggestion
target workload distribution
Improvement
target workload distribution
Consolidation
Scheduling
Task Allocation
Migration
Improvement
target workload distribution
Improvement
target workload distribution

Fig. 2. The framework for energy-efficient scheduling proposed in this paper. An initial suggestion for the target workload distribution is suggested by a domain specific expert. Task allocation is conducted in such that the workload is distributed as similar as possible to the target workload distribution. The target workload distribution is improved over time by machine learning techniques.

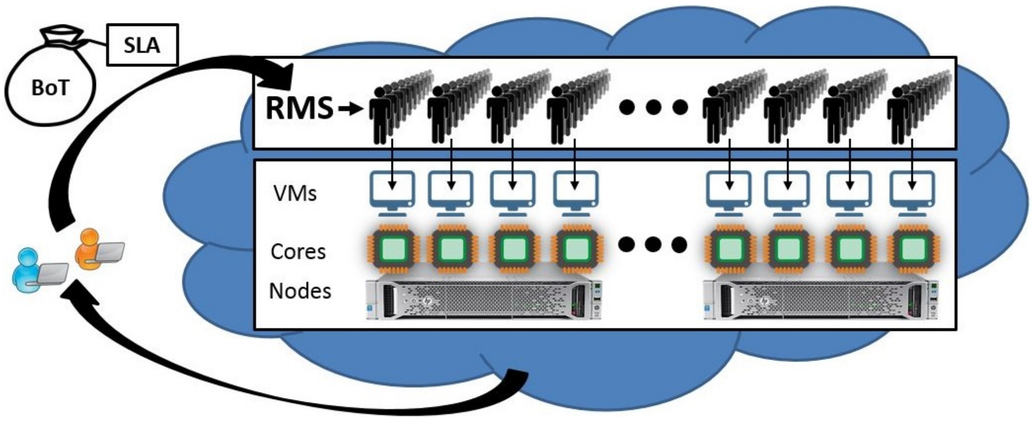

Fig. 3. The system and application model considered in this paper. Clients submit BoT applications to a data center. The data center consists of nodes with multiple cores. Each core operates a single VM. The RMS manages queues on each VM to store tasks in a waiting list while the VM is occupied.

cores. As a consequence, each VM has full access to the physical resources of the underlying core (apart from small amounts dedicated to components such as the Virtual Machine Manager). The VMs on M_i are denoted by $V_{i\ell}$ with $1 \leq \ell \leq \gamma_i$ and have a CPU capacity of ω_i each. Note that VMs on the same node have identical CPU capacities as we assume that cores on the same node are identical.

Jobs $j = 1, 2, \ldots$ are submitted to the system over time. In this work, each job j is a CPU intensive Bag-of-Tasks (BoT) application. We focus on CPU intensive jobs as it allows for considering only CPU as a computational resource. This enables us to focus on the capability of our algorithms to provide good scheduling results. Without loss of generality, we assume that at each point of time at most one job is submitted. In case several jobs are submitted by the exact same time, jobs are stored in a queue to be scheduled shortly one after another. In the following we consider a single job j and omit indices referring to j. The notation is summarized in Table I.

Job j consists of a set of n independent tasks T_k with $1 \leq k \leq n$. The release time r denotes the time at which j is submitted to the system and the deadline d denotes the time at which all tasks of j have to be completed. If at least one task does not finish by d, then the SLA agreed with the customer who submitted j, is violated and j gets rejected. The processing time of T_k on M_i is denoted by p_{ik} and we assume that it can be predicted at the time j is submitted to the system. These predictions can be obtained through historical analysis, predicting techniques or a combination of them as demonstrated in [25]. Note that p_{ik} is only a predicted value and not the actual processing time which is unknown at time r. Since all cores are assumed to be identical, the processing time depends on the node M_i only rather than on the individual core. A job is completed once all its tasks are completed. The scheduler has to assign tasks to VMs such that the common deadline is respected. Tasks assigned to the same VM are stored in a queue and are executed in a first-in-first-out (FIFO) manner. As a VM may be accessed by applications of different customers, a component managing these accesses is required. To this end, the virtualized IaaS Cloud data center also supports a PaaS layer that provides a

TABLE I
SUMMARY AND EXPLANATION OF NOTATION USED IN THIS PAPER

Notation	Description
m	number of nodes in the data center
j	a job submitted as a BoT application
n	number of tasks of job j
M_i	a node in the data center
γ_i	number of cores/VMs of M_i
$V_{i\ell}$	a VM of $M_i, 1 \leq \ell \leq \gamma_i$
ω_i	CPU capacity of each core of M_i
ρ_i	target utilization fraction for M_i
r	time at which j is submitted
T_k	a task of job $j, 1 \leq k \leq n$
d	deadline at which all tasks of j have to be completed
p_{ik}	estimated processing time of T_k on $V_{i\ell}$
$\delta_{i\ell}$	estimated remaining busy time of $V_{i\ell}$ at time r
$\pi_{i\ell}$	weight of $V_{i\ell}$
$\mathcal{V}$	set of VMs
$\mathcal{V}_h$	set of VMs with hth lowest weight
$U_i(t)$	CPU utilization of M_i at time t
$\overline{U}_i$	CPU utilization averaged over the time horizon
$\mathcal{U}(t)$	data center CPU utilization at time t
$\overline{\mathcal{U}}$	data center CPU utilization averaged over the time horizon
f_{hk}	desirability of assigning T_k to V_h
$[q]$	$[q] = \{1, \ldots, q\}, q \in \mathbb{N}$

Resource Management System (RMS). The RMS coordinates job execution of various users in the data center. When j is submitted to the system, a VM $V_{i\ell}$ may still be busy executing tasks from previous jobs. We assume that we can predict the remaining busy time of $V_{i\ell}$, i.e. the time needed until all tasks currently hold in the queue of $V_{i\ell}$ are executed. The remaining busy time of $V_{i\ell}$ at time r is denoted by $\delta_{i\ell}$. As for p_{ik}, the value $\delta_{i\ell}$ is only a prediction.

V. ALGORITHMIC SOLUTIONS

In this section we present algorithms to solve the optimization problem obtained from the framework. The approach is based on assigning weights to VMs. Let $\mathcal{V} = \{V_{i\ell} \mid 1 \leq i \leq m, 1 \leq \ell \leq \gamma_i\}$ denote the set of VMs. We chose a node M_{i^*} such that $i^* := \arg\max_{1 \leq i \leq m} \rho_i$. A *preliminary* weight $\pi'_{i\ell} := \lceil 10\ell \frac{\rho_{i^*}\omega_i}{\rho_i \omega_{i^*}} \rceil$ is first defined for $V_{i\ell}$. The number of different values of preliminary weights may be large. To increase the number of identical values, we define the weights of VMs by categorizing preliminary weights into ten groups:

$$\pi_{i\ell} := \min_{0 \leq \alpha \leq 9} \left\{ \pi = \pi'_{\min} + \frac{\alpha}{9}(\pi'_{\max} - \pi'_{\min}) \mid \pi \geq \pi'_{i\ell} \right\}.$$

The remainder of this section introduces algorithms that aim at minimizing the metrics introduced in Section III-A. In Section V-A this is attempted by approximately solving a generalized assignment problem. The approach presented in Section V-B iteratively assigns subsets of tasks to subsets of VMs.

A. Generalized Assignment Problem

An Integer Linear Programming (ILP) formulation of our scheduling problem is presented in the following:

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} \sum_{\ell=1}^{\gamma_i} \sum_{k=1}^{n} \pi_{i\ell} p_{ik} x_{i\ell k} \\ \text{s.t.} \quad & \sum_{k=1}^{n} p_{ik} x_{i\ell k} \leq d - r - \delta_{i\ell} \quad && i \in [m], \\ & && \ell \in [\gamma_i] \\ & \sum_{i=1}^{m} \sum_{\ell=1}^{\gamma_i} x_{i\ell k} = 1 && k \in [n] \\ & x_{i\ell k} \in \{0,1\} && i \in [m], \\ & && \ell \in [\gamma_i], \\ & && k \in [n], \end{aligned} \tag{1}$$

where the notation $[q] := \{1, 2, \ldots, q\}$ for $q \in \mathbb{N}$ is used. If $x_{i\ell k} = 1$, then T_k is assigned to $V_{i\ell}$. If $x_{i\ell k} = 0$ for all i and ℓ, then task T_k is not assigned to any VM and j is rejected as a consequence. The objective function in (1) minimizes the total processing time of assigned tasks weighted by the weight of the operating VM. The first restriction ensures that tasks respect the common deadline and the remaining busy time of the assigned VM. The second restriction ensures that each task is assigned to at most one VM. In the following we show that the ILP in (1) can be reformulated as a Generalized Assignment Problem (GAP) [26], [27] which is of the following form:

$$\begin{aligned} \min \quad & \sum_{h=1}^{M} \sum_{k=1}^{n} a_{hk} x_{hk} \\ \text{s.t.} \quad & \sum_{k=1}^{n} b_{hk} x_{hk} \leq c_h \quad && h \in [M] \\ & \sum_{h=1}^{M} x_{hk} = 1 && k \in [n] \\ & x_{hk} \in \{0,1\} && h \in [M], k \in [n]. \end{aligned} \tag{2}$$

To get from (1) to (2), we renumber VMs and ignore actual nodes. Formally, we replace indices $i \in [m]$ and $\ell \in [\gamma_i]$ by index $h(i, \ell) := m(i-1) + \ell$ for $h \in [M]$ and $M := \sum_{i=1}^{m} \gamma_i$. Then, for $\pi_h := \pi_{i\ell}$ and $\delta_h := \delta_{i\ell}$, we set

$$a_{hk} := \pi_h p_{i(h),k}, \qquad b_{hk} := p_{i(h),k} \qquad \text{and} \qquad c_h := d - \delta_h,$$

where $i(h)$ is the unique node index $i \in [m]$ that corresponds to index $h \in [M]$. With above substitutions we have completely transformed Problem (1) into the GAP formulated in (2). As the GAP is well-known to be NP-hard, a heuristic approach is necessary. In this work, we use a well-established heuristic framework called MTHG [28] by Martello and Toth. Note that this is not a “ready-to-use” algorithm as in the heuristic framework there still needs to be chosen measures of *desirability*. The desirability f_{hk} is a heuristic indicator of how beneficial it is to assign T_k to V_h. A pseudocode of the MTHG algorithm is presented in Algorithm 1.

The heuristic consists of two phases. In the first phase, unassigned tasks are iteratively considered to determine task T_{k^*} with maximum difference between the largest and second largest desirability f_{hk} for $h \in [M]$. Task T_{k^*} is then assigned

Algorithm 1: MTHG

```
Input  : Bag-of-Tasks $j$, $a_{hk}, b_{hk}, c_h$
Output: Schedule for tasks of $j$ or decision of rejection
1  $\mathcal{T} := \{T_1, \ldots, T_n\}$;          // unassigned tasks
2  $\bar{c}_h := c_h$;        // remaining available time
   /* Phase 1: find feasible solution    */
3  while $\mathcal{T} \neq \emptyset$ do
4      $d^* = -\infty$;
5      foreach $T_K \in \mathcal{T}$ do
6          $F_k := \{V_h \in \mathcal{V} \mid b_{hk} \leq \bar{c}_h\}$;
7          if $F_k = \emptyset$ then
8              Return infeasible;
9          Choose $h'$ so that $f_{h'k} = \max\{f_{hk} \mid h \in F_k\}$;
10         if $|F_k| = 1$ then
11             $d = +\infty$;
12         else
13             $d = f_{h'k} - \max\{f_{hk} \mid h \in F_k \setminus \{h'\}\}$;
14         if $d > d^*$ then
15             $d^* = d, h^* = h', k^* = k$;
16     end
17     Assign $T_{k^*}$ to $V_{h^*}$;
18     $\mathcal{T} = \mathcal{T} \setminus \{T_{k^*}\}$;          // $T_{k^*}$ now assigned
19     $\bar{c}_{h^*} = \bar{c}_{h^*} - b_{h^*k^*}$;  // update availability
20 end
   /* Phase 2: improve solution quality  */
21 for $k \in \{1, \ldots, n\}$ do
22     Move $T_k$ to the VM that reduces the objective
         function the most;
23 end
```

to the VM that maximizes f_{hk^*}. In the second phase, the solution is improved by local improvement exchanges.

For our problem we choose the desirability as $f_{hk} = -\frac{\pi_h}{\bar{c}_h \omega_{i(h)}}$, where $\bar{c}_h$ is as in the MTHG algorithm. Lower weights π_h respectively higher CPU capacities $\omega_{i(h)}$ or values of $\bar{c}_h$ are empirically seen as beneficial for assigning T_k to V_h, thus contributing to a higher desirability f_{hk}.

B. Iterative Algorithm using LPT or FFD

In this section we present an algorithm that consists of a general procedure that iteratively assigns subsets of tasks in phases. In each phase, unscheduled tasks of job j are attempted to be assigned to a subset of VMs using an algorithm $\mathcal{A}$. We present two choices for $\mathcal{A}$ later in this section. In the first phase, the set of VMs $\mathcal{V}_0$ with lowest weight is considered and tasks of j are attempted to be assigned to VMs in $\mathcal{V}_0$. If not all tasks were scheduled, then in a second phase the remaining tasks are attempted to be assigned to the set of VMs with second-lowest weight $\mathcal{V}_1$. This procedure continues until all tasks of j are assigned or until $\mathcal{V}_9$ (the set of VMs with highest weights) was investigated with at least one unassigned task remaining. In the latter case j gets rejected. A pseudocode for the General Procedure is presented in Algorithm 2. We present two choices for $\mathcal{A}$ and justify their relevancy.

Longest Processing Time rule (LPT) "Iteratively consider the VM that becomes available the soonest. Assign the unscheduled task with longest processing time to the VM. Repeat until all tasks are scheduled or until a task cannot be assigned to any VM."

Algorithm 2: General Procedure

```
Input  : Bag-of-Tasks $j$, algorithm $\mathcal{A}$
Output: Schedule for tasks of $j$ or decision of rejection
1 $\alpha := 0$;
2 while not all tasks $T_{jk}$ of $j$ are scheduled do
3     if There exists a VM in $\mathcal{V}_\alpha$ and $\alpha \leq 9$ then
4         Schedule remaining tasks to VMs in $\mathcal{V}_\alpha$ using
            algorithm $\mathcal{A}$;
5     else
6         Reject $j$;
7     end
8     $\alpha = \alpha + 1$;
9 end
```

We choose the LPT-rule for two reasons. First, the LPT is straightforward to implement and only requires minimal computation time to execute. Second, the LPT-rule has proven approximation ratios if the makespan is considered as an objective function. Algorithm $\mathcal{A}$ aims to assign as many tasks as possible respecting deadline d. This feasibility-problem with respect to a common deadline is equivalent to a makespan-minimization problem without a deadline [29]. Therefore, we believe that the mathematically proven approximation-ratios are still a valuable indicator for appropriate algorithms. The approximation-ratio for the LPT-rule is $\frac{4}{3} - \frac{1}{3m}$ in case of identical nodes [30] and $\frac{19}{12}$ in case of nodes of different speeds [31]. Moreover, the LPT-rule is proven to be asymptotically optimal [32], [33], i.e. the LPT-rule will actually deliver the optimal solution if the number of tasks is sufficiently large.

First Fit Decreasing rule (FFD) "Iteratively consider the VM that becomes available the latest. Assign the unscheduled task with longest processing time to the VM if feasible. Repeat until all tasks are scheduled or until a task cannot be assigned to any VM."

The difference between the LPT-rule and the FFD-rule is that the LPT-rule first considers VMs in order of the times at which they become free next, whereas in contrast the FFD-rule first considers VMs in the reversed order. As a consequence, the LPT-rule acts as a load balancer, whereas the FFD-rule aims at reducing the number of required VMs. It is proven [34] that the FFD-rule does not occupy more than $\frac{11}{9}\mathrm{OPT}(I) + \frac{6}{9}$ VMs, where $\mathrm{OPT}(I)$ is the minimum number of required VMs in order to schedule all tasks.

VI. Performance Evaluation

We conduct a performance simulation using the SEED simulator [35]. We choose the initial target workload distribution inspired by the simulation results from [5] which is illustrated in Figure 1. Note that although the intention of the framework is to obtain the recommended target workload distribution by the scientist, we feel that it is reasonable to take the workload distribution from [5] as a "simulated recommendation".

The data center consists of 24 nodes. The number of cores/VMs on each node is randomly chosen from $\{2, 4, 8\}$. Each VM is assigned a CPU capacity ω_i which is randomly chosen from $\{1, 1.25, 1.5\}$ with the restriction that VM on the same node are assigned the same CPU capacity. The quantities for CPU capacity have been normalized to make comparison and interpretation easier.

Probability distributions and parameters in the following have been chosen as in [24]. Jobs are submitted according to a Weibull distribution with parameters $(4.25, 7.86)$ which is in accordance with findings from the comprehensive analysis of workload characteristics conducted in [25]. A job consists of 2^W tasks, where W follows a Weibull distribution with parameters $(1.76, 2.11)$. The actual execution time P_{ik} of task T_k assigned to a VM on M_i is given by $P_{ik} = \frac{2^X}{\omega_i}$ minutes, where X follows a normal distribution with mean 2.73 and standard deviation 6.1. The deadline d is modeled by $d = r+2^{\mathrm{E}(X)}T$, where T follows a normal distribution with mean 9 and standard deviation 2.2. The estimated processing time p_{ik} and the estimated remaining busy time $a_{i\ell}$ are modeled by their respective expected values. The time horizon considered in the simulation is 24 hours.

We evaluate our MTHG and our General Procedure algorithm with LPT- and FFD rule as subroutines. The evaluation is conducted by a comparison against a load balancing algorithm that assigns tasks in a round robin manner as well as against a probabilistic asymmetric load balancing algorithm that assigns a task to $V_{i\ell}$ with probability $\frac{\rho_i}{\gamma_i}$ if feasible. This way, M_i is expected to be assigned a fraction ρ_i of the data center workload over time. The entire simulation is conducted five times. Averaged values for each performance metric are presented in Table II. An illustration for the performance on an individual node with 8 VMs is given in Figure 4.

The results show that our algorithms outperform both the load balancing and asymmetric load balancing algorithm for each of the three objectives. As expected, the load balancing algorithm has the worst performance with an average relative deviation (RD) of 187% and an average relative variance (RVar) that is 9.66 times higher as desired. The asymmetric load balancing algorithm improves upon the load balancing algorithm by taking also the target utilization level into consideration for the scheduling decision. This significantly reduces the average RD to 142% with the cost of a slightly increased average RVar. Our proposed algorithms further improve upon the asymmetric load balancing algorithm. They not only take into account the target utilization levels, but also other characteristics such as the CPU capacity or estimates of task execution times. As a result, our algorithms achieve a further reduction of the RD to a level of between 86% and 118% as well as a reduction of the RVar to between 6.24 and 7.47. Similar conclusions can be drawn for the IBL metric.

Finally we highlight that although e.g. a RD of about 86% may appear substantive, however our particular problem contains many constraints making it a highly restrictive problem. As a consequence, solutions with a very low RD or RVar value are not likely to exist. This is further fortified by very high RD and RVar values obtained from (asymmetric) load balancing.

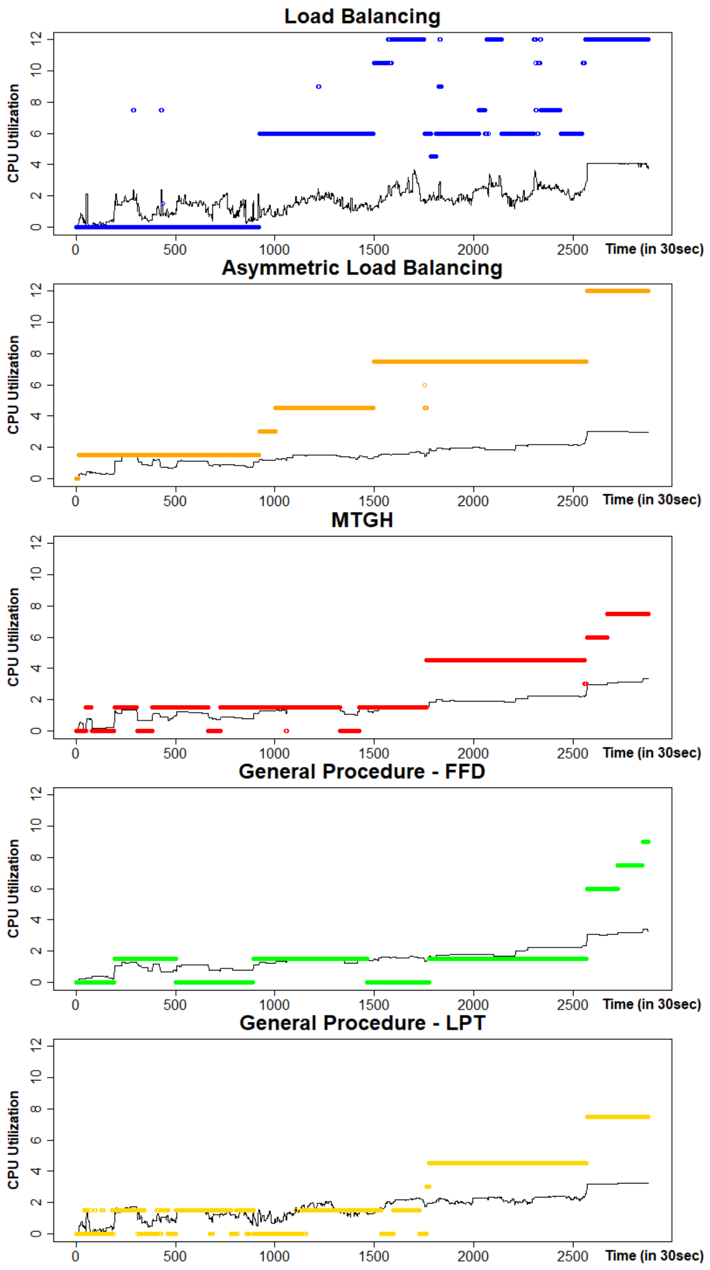

Fig. 4. Performance illustration on a node with 8 VMs. The black thin graph is the desired utilization level of the node, whereas the thick line in color is the actual utilization level obtained by the corresponding scheduling algorithm.

TABLE II
AVERAGE PERFORMANCE RESULTS

	IBL	RD	RVar
Load Balancing	1.29×10^8	1.87	9.66
Asymmetric Load Balancing	8.69×10^7	1.42	10.6
MTHG (GAP)	$\mathbf{4.70 \times 10^7}$	**0.86**	**6.24**
LPT - Procedure	7.98×10^7	1.06	7.39
FFD - Procedure	7.08×10^7	1.18	7.47

VII. Conclusion and Future Work

In this paper we have proposed an infrastructure independent framework for energy-efficient scheduling in Cloud data centers that does not require complex modeling of energy. The framework consists of several components with the task allocation component being the focus of this work. In the task allocation component, a scheduler is required to assign workload to nodes such that the workload distribution gets as similar as possible to a given target workload distribution. We have proposed several algorithms and conducted comprehensive performance evaluation using simulation. The simulation results demonstrate that algorithms yield a reduction with respect to the (asymmetric) load balancing algorithm by at least $16.9\% = 1 - \frac{1.18}{1.42}$ for the relative deviation and a reduction by at least $22.67\% = 1 - \frac{7.47}{9.66}$ for the relative variance.

As valuable insights into the task allocation component have been found in this work, future work includes studying the other components of the framework. Once each component is sufficiently understood we plan to combine components together in order to conduct a performance simulation for the entire framework. We plan to conduct practical experiments using real infrastructure to evaluate it's effectiveness, and engage with additional domain specific experts in Mechanical Engineering to provide an initial target workload distribution required in the framework.

Acknowledgments

This work was supported by the EPSRC (EP/P031617/1).

References

[1] P. Corcoran and A. Andrae, "Emerging trends in electricity consumption for consumer ict," *National University of Ireland, Galway, Connacht, Ireland, Tech. Rep*, 2013.

[2] J. Koomey, "Growth in data center electricity use 2005 to 2010," *A report by Analytical Press, completed at the request of The New York Times*, vol. 9, 2011.

[3] R. Zhou, Z. Wang, C. E. Bash, and A. McReynolds, "Data center cooling management and analysis-a model based approach," in *Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM), 2012 28th Annual IEEE*. IEEE, 2012, pp. 98–103.

[4] J. D. Moore, J. S. Chase, P. Ranganathan, and R. K. Sharma, "Making scheduling" cool": Temperature-aware workload placement in data centers." in *USENIX annual technical conference, General Track*, 2005, pp. 61–75.

[5] X. Li, P. Garraghan, X. Jiang, Z. Wu, and J. Xu, "Holistic virtual machine scheduling in cloud datacenters towards minimizing total energy," *IEEE Transactions on Parallel and Distributed Systems*, 2017.

[6] Y. Fulpagare and A. Bhargav, "Advances in data center thermal management," *Renewable and Sustainable Energy Reviews*, vol. 43, pp. 981–996, 2015.

[7] A. Mousavi, V. Vyatkin, Y. Berezovskaya, and X. Zhang, "Towards energy smart data centers: Simulation of server room cooling system," in *Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on*. IEEE, 2015, pp. 1–6.

[8] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *The Journal of Supercomputing*, vol. 60, no. 2, pp. 268–280, May 2012. [Online]. Available: https://doi.org/10.1007/s11227-010-0421-3

[9] A. Beloglazov and R. Buyya, "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers." in *MGC@ Middleware*, 2010, p. 4.

[10] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," in *Proceedings of the 2008 conference on Power aware computing and systems*, vol. 10. San Diego, California, 2008, pp. 1–5.

[11] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[12] E. G. Coffman Jr, M. R. Garey, and D. S. Johnson, "Approximation algorithms for bin-packingan updated survey," in *Algorithm design for computer system design*. Springer, 1984, pp. 49–106.

[13] A. Lodi, S. Martello, and D. Vigo, "Recent advances on two-dimensional bin packing problems," *Discrete Applied Mathematics*, vol. 123, no. 1, pp. 379–396, 2002.

[14] A. J. Younge, G. Von Laszewski, L. Wang, S. Lopez-Alarcon, and W. Carithers, "Efficient resource management for cloud computing environments," in *Green Computing Conference, 2010 International*. IEEE, 2010, pp. 357–364.

[15] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proceedings of the 2010 10th IEEE/ACM international conference on cluster, cloud and grid computing*. IEEE Computer Society, 2010, pp. 826–831.

[16] T. Guérout, T. Monteil, G. Da Costa, R. N. Calheiros, R. Buyya, and M. Alexandru, "Energy-aware simulation with dvfs," *Simulation Modelling Practice and Theory*, vol. 39, pp. 76–91, 2013.

[17] S. Albers, *Algorithms for Energy Saving*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 173–186.

[18] A. Wierman, L. L. H. Andrew, and M. Lin, "Speed scaling: An algorithmic perspective."

[19] Z. Song, X. Zhang, and C. Eriksson, "Data center energy and cost saving evaluation," *Energy Procedia*, vol. 75, pp. 1255–1260, 2015.

[20] S. Shrivastava, B. Sammakia, R. Schmidt, and M. Iyengar, "Comparative analysis of different data center airflow management configurations," *ASME Paper No. IPACK2005-73234*, 2005.

[21] B. Durand-Estebe, C. Le Bot, J. N. Mancos, and E. Arquis, "Data center optimization using pid regulation in cfd simulations," *Energy and Buildings*, vol. 66, pp. 154–164, 2013.

[22] J. Summers, N. Kapur, and H. Thompson, "Design of data centre rack arrangements using open source software," in *Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM), 2013 29th Annual IEEE*. IEEE, 2013, pp. 45–51.

[23] S. Shrivastava, B. Sammakia, R. Schmidt, and M. Iyengar, "Comparative analysis of different data center airflow management configurations," *ASME Paper No. IPACK2005-73234*, 2005.

[24] R. N. Calheiros and R. Buyya, "Energy-efficient scheduling of urgent bag-of-tasks applications in clouds through dvfs," in *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*. IEEE, 2014, pp. 342–349.

[25] I. S. Moreno, P. Garraghan, P. Townend, and J. Xu, "Analysis, modeling and simulation of workload patterns in a large-scale utility cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 208–221, 2014.

[26] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer-Verlag Berlin Heidelberg, 2004.

[27] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. New York, NY, USA: John Wiley & Sons, Inc., 1990.

[28] ——, "An algorithm for the generalized assignment problem," *Operational research*, vol. 81, pp. 589–603, 1981.

[29] P. Brucker, *Scheduling algorithms*. Springer-Verlag Berlin Heidelberg, 2007.

[30] O. Braun, F. Chung, and R. Graham, "Bounds on single processor scheduling with time restrictions," in *Proceedings of the 14th International Conference on Project Management and Scheduling*, 2014, pp. 48–51.

[31] G. Dobson, "Scheduling independent tasks on uniform processors," *SIAM J. Comput.*, vol. 13, no. 4, pp. 705–716, Nov. 1984. [Online]. Available: http://dx.doi.org/10.1137/0213044

[32] E. G. Coffman, Jr. and R. Sethi, "A generalized bound on lpt sequencing," in *Proceedings of the 1976 ACM SIGMETRICS Conference on Computer Performance Modeling Measurement and Evaluation*, ser. SIGMETRICS '76. New York, NY, USA: ACM, 1976, pp. 306–310. [Online]. Available: http://doi.acm.org/10.1145/800200.806205

[33] B. Chen, "A note on lpt scheduling," *Operations Research Letters*, vol. 14, no. 3, pp. 139 – 142, 1993. [Online]. Available: http://www.sciencedirect.com/science/article/pii/016763779390024B

[34] G. Dósa, R. Li, X. Han, and Z. Tuza, "Tight absolute bound for first fit decreasing bin-packing: Ffd (l) 11/9opt (l)+ 6/9," *Theoretical Computer Science*, vol. 510, pp. 13–61, 2013.

[35] P. Garraghan, D. McKee, X. Ouyang, D. Webster, and J. Xu, "Seed: A scalable approach for cyber-physical system simulation," *IEEE Transactions on Services Computing*, vol. 9, no. 2, pp. 199–212, 2016.