Ulrich Lampe, Melanie Siebenhaar, Ronny Hans, Dieter Schuller, Ralf Steinmetz:
Let the Clouds Compute: Cost-Efficient Workload Distribution in Infrastructure Clouds.
In: Kurt Vanmechelen, Jörn Altmann, Omer Rana: Proceedings of the 9th International Conference on Economics of Grids, Clouds, Systems, and Services (GECON 2012), vol. Lecture Notes in Computer Science, no. 7713, p. 91-101, Springer, November 2012. ISBN 978-3-642-35193-8.

Let the Clouds Compute: Cost-Efficient Workload Distribution in Infrastructure Clouds

Ulrich Lampe, Melanie Siebenhaar, Ronny Hans, Dieter Schuller, Ralf Steinmetz

Multimedia Communications Lab (KOM), TU Darmstadt, Germany
Ulrich.Lampe@kom.tu-darmstadt.de
http://www.kom.tu-darmstadt.de/

Abstract. With cloud computing, a virtually inexhaustible pool of computing capacity has become available to IT users. However, given the large number of Infrastructure as a Service offers with differing pricing options, the cost-efficient distribution of workloads poses a complex challenge. In this work-in-progress paper, we formally describe the *Cloud-oriented Workload Distribution Problem* and propose an exact as well as a heuristic optimization approach. Through an evaluation that is based on realistic data from the cloud market, we examine the performance and practical applicability of these approaches.

Keywords: Cloud Computing, Infrastructure, Workload, Distribution, Deployment, Offline, Optimization, Exact, Heuristic

1 Introduction

With the advent of cloud computing, Information Technology (IT) services have increasingly become commodities over the last few years, resulting in the vision of "IT as the fifth utility" [1]. Combined with aspects such as the elimination of upfront investments and high scalability, the idea to lease computing capacity – rather than provide it in-house – increasingly gains in appeal.

In this work, we assume that a user aims to deploy a *workload* onto leased cloud infrastructure at minimal cost. We define a workload as a set of multiple computational jobs, which are executed on *Infrastructure as a Service* (IaaS) capacities in the form of *Virtual Machines* (VMs). The decision where to place these jobs is complicated by two main factors: First, VMs are discrete compute units that only provide a limited supply of certain resources, such as processor power or local storage space. Second, billing schemes commonly differ between the various cloud providers and involve different price components, such as periodical VM leasing fees and usage-based network traffic fees.

In the work at hand, we refer to this challenge as *Cloud-oriented Workload Distribution Problem* (CWDP). In the following Section 2, the problem is defined in detail. In Section 3, we present exact and heuristic optimization approaches to solve the CWDP. A quantitative evaluation of both approaches is described in Section 4. Section 5 provides an overview of related work. Lastly, Section 6 concludes the paper with a summary and an outlook on future work.

2 Problem Statement

As mentioned before, we address the so-called CWDP in the work at hand. We assume that a user has specified a workload, which he/she aims to deploy onto leased cloud infrastructure in the form of VMs. The workload consists of individual computational jobs with given durations, i. e., the initiation and termination times of all jobs are known in advance. Computational jobs may, e. g., represent distributed application components or tasks in a scientific algorithm.

Each job exhibits certain resource demands throughout its execution, e. g., in terms of processor power or network bandwidth. Jobs cannot be split among different VM instances; however, multiple jobs may be combined on one VM instance. In addition, all jobs are non-preemptable, i. e., they have to be continuously executed.

Within the cloud market, different VM types are available to the user from multiple IaaS providers, such as Amazon[1] or Microsoft[2]. Each instance of a VM type supplies a different quantity of resources, e. g., processor power, and imposes a certain usage fee per *fixed-length* leasing period. In addition, surcharges for each unit of resource consumption, e. g., network traffic, may apply.

The objective of the user consists in minimizing his/her total leasing cost. As a constraint, the resource demands of all jobs within the workload have to be satisfied by corresponding resource supplies.

3 Optimization Approaches

In this section, we first introduce formal notations, which subsequently permit to define the CWDP as mathematical optimization model and infer an exact optimization approach. Due to the computational complexity of such exact approach, we additionally present a heuristic optimization approach.

3.1 Formal Notations

In order to map the CWDP into a formal optimization model, we introduce a few formal notations. To begin with, let $J \subset \mathbb{N}$ denote the set of jobs that comprise the workload. For each job $j \in J$, its initiation time $TI_j \in \mathbb{N}$, termination time $TT_j \in \mathbb{N}$, and corresponding duration $D_j \in \mathbb{N}$ are given. The jobs exhibit computational demands, which refer to the set of resource types $R \subset \mathbb{N}$. Specifically, $RD_{j,r}$ denotes the resource demand that job j imposes on a VM instance with respect to the resource type $r \in R$.

The available VM types are given by the set $V \subset \mathbb{N}$. The maximum number of concurrently available instances of VM type $v \in V$, as specified by the respective IaaS provider, correspond to $m_v \in \mathbb{N}$. We further define $RS_{v,r}$ as the supply of resource type r per individual instance of VM type v. $P_v \in \mathbb{N}$ denotes the fixed

[1] `http://aws.amazon.com/ec2/`

[2] `http://www.windowsazure.com/`

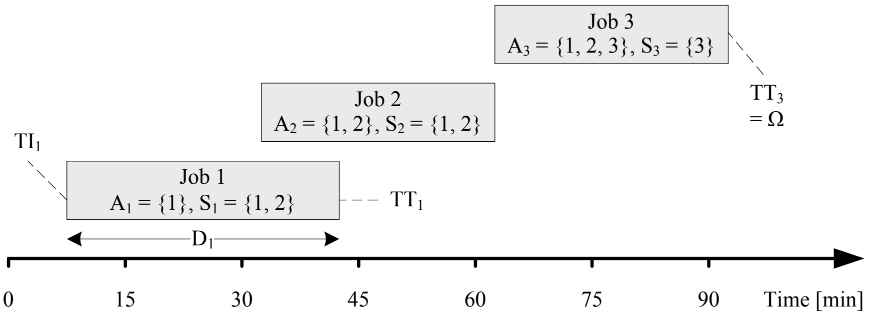


Fig. 1. Example workload with selected formal notations.

length of a single leasing period for each VM type v, for which a fixed usage fee of $CF_v \in \mathbb{R}^+$ will be charged. In addition, $CV_{v,r} \in \mathbb{R}^+$ denotes the cost of resource type r per used resource and time unit.

Based on the above notations, we further define $\Omega = \max_{j \in J}(TT_j)$ as the maximum termination time among all jobs in the workload. In addition, $A_j = \{j' \in J \mid TI_{j'} \leq TI_j\}$ denotes the jobs that are initiated prior to a given job j. Likewise, $S_j = J \setminus \{j' \in J \mid TT_{j'} < TI_j \vee TI_{j'} > TT_j\}$ specifies jobs that are executed simultaneously with j. Please note that both A_j and S_j include j itself (an example is provided in Figure 1).

3.2 Exact Optimization Approach

In order to compute an exact, i. e., cost-minimal, solution to the CWDP, we map the problem statement from Section 2 into its mathematical equivalent. The result is provided in Model 1. Prior to a detailed explanation, we introduce two concepts that are relevant to the understanding of the model.

First, we observe that the lease of a VM instance will always start with the initiation of a job. For a specific VM type v, the lease may consecutively be renewed after one leasing period of the fixed length P_v. The lease will, at the latest, terminate once the final computational job has finished. Thus, based on a given job j, we define a set of (potential) leasing instants $L_{v,j} = \{TI_j, \ldots, TI_j + k \times P_v\}$, where $k = \lfloor(\Omega - TI_j)/P_v\rfloor$. For all jobs in the workload, the set of leasing instants is given by $L_v = \bigcup_{j \in J} L_{v,j}$ accordingly.

For reasons of convenience, we additionally define the set of neighboring leasing instants $N_v(t) = \{l \in L_v \mid l > t - P_v \wedge l \leq t\}$, which lie within the length of one leasing period P_v before a specified time instant $t \in \mathbb{N}$.

Second, a VM instance must be leased throughout the complete duration of the jobs that have been assigned to it. The existence of such active lease can be verified at certain checkpoints, which temporally coincide with the potential leasing instants that may be relevant to a job. Based on a given job j and

Model 1 Exact Approach for Cloud-oriented Workload Distribution

$$\text{Min. } TC(x,y) = \sum_{v \in V, i \in I_v, l \in L_v} y_{v,i,l} \times CF_v \tag{1}$$

$$+ \sum_{j \in J, v \in V, i \in I_v} x_{j,v,i} \times D_j \times RD_{j,r} \times CV_{v,r}$$

subject to

$$\sum_{l \in N_v(c)} y_{v,i,l} \geq x_{j,v,i} \quad \forall j \in J, v \in V, i \in I_v, c \in C_{v,j} \tag{2}$$

$$\sum_{v \in V, i \in I_v} x_{j,v,i} = 1 \quad \forall j \in J \tag{3}$$

$$\sum_{j' \in S_j} x_{j',v,i} \times RD_{j',r} \leq RS_{v,r} \quad \forall v \in V, i \in I_v, j \in J, r \in R \tag{4}$$

$$\sum_{l' \in N_v(l)} y_{v,i,l'} \leq 1 \quad \forall v \in V, i \in I_v, l \in L_v \tag{5}$$

$$I_v = \{1, \ldots, \min(\max_{j \in J}(|S_j|), m_v)\} \tag{6}$$

$$I_v \subset \mathbb{N}$$

$$x_{j,v,i} \in \{0,1\} \quad \forall j \in J, v \in V, i \in I_v \tag{7}$$

$$y_{v,i,l} \in \{0,1\} \quad \forall v \in V, i \in I_v, l \in L_v \tag{8}$$

the previously defined leasing instants, we define the set of checkpoints as $C_{v,j} = \{TI_j, TT_j\} \cup \bigcup_{j' \in A_j} \{l \in L_{v,j'} \mid l > TI_j \wedge l < TT_j\}$.

Based on these concepts, the optimization model can be explained in further detail: To begin with, Equation 1 defines the objective, which consists in minimizing the total leasing cost of the cloud infrastructure. The total cost comprises two components, namely the periodical leasing fees for the VM instances and the additional charges for resource consumption.

For that matter, the decision variable $x_{j,v,i}$ indicates whether a job $j \in J$ has been assigned to a VM instance of type $v \in V$ with the running index $i \in I_v$ or not. In a similar manner, the decision variable $y_{v,i,l}$ indicates whether the VM instance of type v with the running index $i \in I_v$ has been leased at the time instant $l \in L_v$ or not. Both x and y are defined as binary variables in Equations 7 and 8. Whereas x is the main decision variable, y can be interpreted as an auxiliary decision variable.

Equation 2 links the two decision variables. More precisely, it defines that if an assignment of a task to a certain VM instance has been made, as indicated by variable x, each checkpoint needs to be matched by a corresponding active lease, as represented by variable y.

Equation 3 ensures that each task is assigned to precisely one VM instance. Equation 4 guarantees that the resource demands of all tasks that are executed simultaneously are met by corresponding resource supplies. Equation 5 ensures that the lease of a VM instance cannot be renewed until the previous lease has expired. Equation 6 defines a set of valid instance indices for each VM type. The definition is based on the notion that, in the worst case, the largest set of simultaneous jobs within the workload, i. e., $max_{j \in J}(|S_j|)$, may be deployed on individual VM instances of the same type; yet, in any case, no more than the maximum number of VM instances, i. e., m_v, can be used.

As can be seen, Model 1 constitutes a special case of a linear program, namely a *Binary Integer Program* (BIP). Such BIP can be solved using well-known methodologies from the field of operations research, such as *branch and bound* [5]. Unfortunately, the computational complexity of such exact methods can be very high. In the worst case, the complexity grows exponentially with the number of decision variables. In the specific case of the CWDP, the worst case complexity corresponds to $\mathcal{O}(2^{|J|^2 \times |V| + |J| \times \sum_{v \in V} L_v})$. Accordingly, an exact solution to the CWDP can most likely not be computed within reasonable time if a workload features multiple jobs, or long job durations that result in a large number of potential leasing instants.

3.3 Heuristic Optimization Approach

In the previous Section 3.2, we have explained that the complexity of computing an exact solution to the CWDP rapidly increases with the number of jobs in the workload. This indicates the need for a heuristic optimization approach, which potentially trades reductions in computation time against possibly sub-optimal solutions.

The heuristic presented in this work is based on an approach that we have previously proposed for the cost-efficient distribution of software services [6]. Because this previous work assumed an online (i. e., at run time), rather than an offline distribution (i. e., at design time), a number of adaptations had to be made. In both cases, however, the principal mechanism is inspired by heuristic solutions to the well-known *knapsack problem* [3].

Our heuristic approach encompasses two phases, *VM packing* and *VM selection*, which are iteratively repeated until a valid solution to the CWDP has been computed. The principle idea is to select a subset of jobs in each iteration and assign it to a new instance of the most cost-efficient VM type. A schematic overview of the complete heuristic in the form of pseudo code is provided in Algorithm 1 and will be explained in the following.

In accordance with Section 3.2, x denotes the main decision variable. Again, $x_{j,v,i}$ indicates whether a job j has been assigned to a VM instance of type v with the running index i or not.

As a preparatory step for the main algorithm, we initialize the current instance index i_v for each VM type v (lines 1–3).

Algorithm 1 Heuristic Approach for Cloud-oriented Workload Distribution

```
 1: for all $v \in V$ do
 2:   $i_v \leftarrow 1$
 3: end for
 4: repeat
 5:   for all $v \in V$ do
 6:     if $i_v \leq m_v$ then
 7:       $PL_v \leftarrow \emptyset$
 8:       repeat
 9:         $\delta \leftarrow$ false
10:         for all $j \in J \setminus PL_v$ do
11:           if CHECKFIT($v, PL_v \cup \{j\}$) then
12:             $u \leftarrow$ COMPUTIL($v, PL_v$)
13:             $u' \leftarrow$ COMPUTIL($v, PL_v \cup \{j\}$)
14:             if $u' > u$ then
15:               $PL_v \leftarrow PL_v \cup \{j\}$
16:               $\delta \leftarrow$ true
17:             end if
18:           end if
19:         end for
20:       until $\delta =$ false
21:     end if
22:   end for
23:   $\hat{u} \leftarrow 0$
24:   $\hat{v} \leftarrow$ null
25:   for all $v \in V$ do
26:     if $i_v \leq m_v$ then
27:       $u_v \leftarrow$ COMPUTIL($v, PL_v$)
28:       if $u_v > \hat{u}$ then
29:         $\hat{u} \leftarrow u_v$
30:         $\hat{v} \leftarrow v$
31:       end if
32:     end if
33:   end for
34:   if $\hat{v} \neq$ null then
35:     for all $j \in PL_{\hat{v}}$ do
36:       $x_{j,v,i_{\hat{v}}} \leftarrow 1$
37:     end for
38:     $J \leftarrow J \setminus PL_{\hat{v}}$
39:     $i_{\hat{v}} \leftarrow i_{\hat{v}} + 1$
40:   end if
41: until $J = \emptyset \vee \hat{v} = \emptyset$
```

In the first phase, *VM packing*, we create a so-called *packing list* for each VM type. A packing list $PL_v \subseteq J$ represents a subset of jobs that have not been assigned yet and would fit onto a new instance of type v. Initially, we assume an empty packing list. Subsequently, we scan the set of jobs, J, in the order of initiation times. If the current job j would additionally fit onto a new VM instance of type v and increase the utility of the packing list, it is added to the packing list. Utility, in this respect, is defined as the ratio between the aggregated durations of all jobs and the corresponding leasing cost. It is computed using the function COMPUTIL. The process of scanning the set of jobs is repeated until no more changes to any packing list could be made, as indicated by the variable δ. This repetition is necessary because a certain job may only increase the utility after a subsequent job has already been added to the packing list (lines 5–22).

In the second phase, *VM selection*, we determine the favorite, i. e., most cost-efficient, VM type, based on the previously created packing lists. In accordance with the previous phase, cost-efficiency is represented by a utility value. The favorite VM type is denoted by $\hat{v}$, with a utility of $\hat{u}$. For each VM type v, we initially assume a utility of zero; VM types for which the maximum number of instances has been reached are excluded from the process. We compute the overall utility value u_v, based on the previously compiled packing list PL_v. If the utility value u_v exceeds the maximal value $\hat{u}$, v is assumed as the new favorite VM type $\hat{v}$ (lines 23–33).

In the following, we check whether a favorite VM type $\hat{v}$ could be identified. This may not be the case if the maximum number of instances of each VM type has been reached, or none of the available VM types is suitable to execute one of the remaining jobs. If a favorite VM type $\hat{v}$ exists, however, we assign all jobs in the packing list $PL_{\hat{v}}$ to a new instance with the index $i_{\hat{v}}$. Finally, we remove the assigned jobs from the set J and increment the instance count $i_{\hat{v}}$ (lines 34–40).

Both phases are iteratively repeated until all jobs have been assigned or no favorite VM type can be identified. In the latter case, a portion of the jobs could not be successfully assigned, thus yielding an invalid solution.

From an analytical point of view, the heuristic has substantial advantages over the exact solution in terms of computational complexity. Specifically, the creation of the packing list is the most complex part; given that the number of resource types is constant and thus negligible, the worst-case complexity of the complete algorithm is polynomial and corresponds to $\mathcal{O}(|J|^4 \times |V|)$.

4 Evaluation

Both previously presented optimization approaches have been prototypically implemented as a Java program, which serves as the basis of our evaluation. For solving the BIP in the exact optimization approach, we apply the *IBM ILOG CPLEX Optimizer*[3].

4.1 Design and Setup

With the evaluation, we aimed to quantitatively assess the performance of the two optimization approaches. Performance, in this respect, is represented by two dependent variables: First, *computation time* represents the scalability of the two approaches and indicates their practical applicability to large-scale CWDPs. Second, *total cost* indicates the solution quality with respect to the objective of cost-efficient workload distribution.

For the evaluation, we created ten classes of CWDPs, each containing 100 individual problems. In classes $A1$ to $A5$, we treated the number of jobs as independent variable, i. e., $|J| \in \{4, 8, 12, 16, 20\}$, assuming a fixed number of VM types, i. e., $|V| = 6$. In classes $B1$ to $B5$, we assumed the number of VM types as independent variable, i. e., $|V| \in \{2, 4, 6, 8, 10\}$, treating the number of jobs as fixed, i. e., $|J| = 12$. Thus, we vary those two variables that have been analytically identified as influential to the computational complexity of both optimization approaches.

For the definition of the VM types, we used the specifications provided by Amazon for its Elastic Compute Cloud (EC2). Each type exhibits specific resource supplies with respect to three resource types (namely processor, memory, and storage), where the consumption is covered by differing hourly leasing fees. In addition, we regarded network traffic – which imposes additional usage-based fees – as fourth resource type, i. e., $|R| = 4$.

[3] http://www.ibm.com/software/integration/optimization/cplex-optimizer/

The workloads were randomly generated by drawing the initiation times and durations of the jobs from the uniform distributions $U(1, 120)$ and $U(1, 60)$ respectively, assuming minutes as time unit. The resource demands of the individual jobs were also randomly drawn, assuming the resource supplies of an Amazon EC2 *Standard Medium* VM as upper limit.

Each CWDP was solved using both optimization approaches, using a desktop computer with an Intel Core 2 Quad Q9450 processor and 8 GB of memory, operating under Microsoft Windows 7. In the process, we imposed a timeout of 300 seconds (i. e., 5 minutes) per problem and optimization approach. Only such problems that could be successfully solved by both approaches within this timeout period were considered in the following analysis.

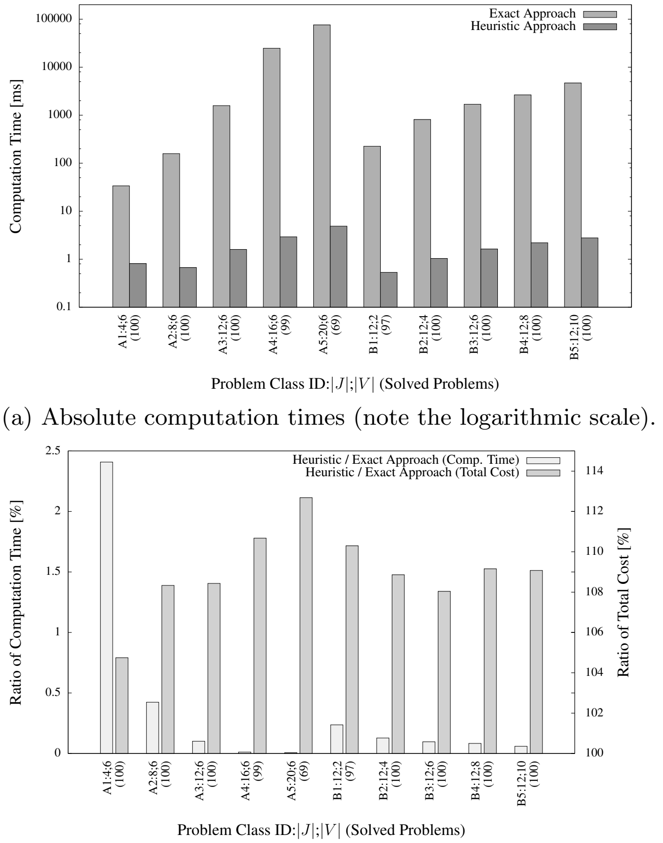
4.2 Results and Discussion

As can be observed in Figure 2a, the exact optimization approach quickly reaches absolute computation times in the magnitude order of seconds (classes $A3$ and $B2$ to $B5$, which involve 12 jobs) or even ten seconds (classes $A4$ and $A5$, which involve 16 and 20 jobs), and also results in various timeouts (specifically for class $A5$). In accordance with our qualitative analysis in Section 3.2, the absolute computation times exhibit an exponential growth with an increasing number of jobs (classes $A1$ to $A5$). The same applies for a growing number of VM types, even though the effect is less pronounced (classes $B1$ to $B5$).

In contrast, for the heuristic approach, the absolute computation times are in the order of milliseconds or below across all evaluated classes. In relative terms, the heuristic achieves reductions in computation time of more than 97.5% compared to the exact approach (cf. Figure 2b). This gap further increases with a growing number of jobs (classes $A1$ to $A5$), which, in accordance with our analysis from Section 3.3, indicates a superior scalability of the heuristic approach. With an increasing number of VM types (classes $B1$ to $B5$), the benefits of the heuristic approach are less accentuated. However, this is also of limited practical relevance, because the number of available VM types will usually be restricted.

Lastly, as it can be seen in Figure 2b, the reduction in computation time is traded against reductions in solution quality. Depending on the problem size, the increase in total cost for the distribution of the workload ranges between approximately 5% and 13%. The gap between both optimization approaches appears to grow with an increasing number of jobs (classes $A1$ to $A5$), but not with a growing number of VM types (classes $B1$ to $B5$). Accordingly, the proposed heuristic should be seen as a first step toward efficiently solving large-scale CWDPs. That is, it primarily provides a valid baseline solution, which should subsequently be refined by a specific improvement procedure.

In summary, our evaluation indicates that the exact optimization approach is solely applicable to small CWDPs involving around 20 jobs in practice. In comparison, the heuristic approach exhibits a more favorable runtime behavior, which renders it potentially suitable for larger CWDPs involving hundreds or thousands of jobs. However, the heuristic also results in notable increases in the total cost of workload distribution. Thus, a future direction of our research will



(a) Absolute computation times (note the logarithmic scale).

(b) Ratios of computation times (left ordinate) and total costs (right ordinate).

Fig. 2. Evaluation results for both optimization approaches.

consist in the development of improvement procedures, which permit enhancements in solution quality while sustaining the low computational complexity of a heuristic solution approach.

5 Related Work

In recent years, the distribution of workloads in cloud and grid environments has been a vivid field of research. Substantial efforts have also been undertaken in the related field of *workflow* distribution. Given space limitations, we only present a brief overview of the most similar works.

To start with, in our own previous work, e. g., [6], we have examined the Software Service Distribution Problem (SSDP). This challenge concerns the distribution of software services within IaaS clouds. Specifically, we have proposed both an exact and a heuristic *online* approach that permits to cost-efficiently allocate software service requests to different VM types at run time. When interpreting software services as computational jobs within a workload, the approach proposed in this work can be seen as a benchmark, which permits to compute a theoretically optimal ex-post, *offline* solution to the SSDP.

Li et al. [7] have studied the distribution of VMs among PMs in order to serve predictable peak loads. Their objective consists in load balancing among the PMs; for that purpose, the authors propose four different algorithms, which are extensively evaluated. While there is a number of similarities between the work by Li et al. and our research, a major difference lies in the objective of cost-efficiency. In addition, we consider fixed-length VM leasing periods, different price components, and multiple resource types in the distribution process.

Genez et al. [4] have examined the problem of cost-efficient workflow deployment in hybrid clouds under consideration of service level agreements. They propose an exact optimization approach based on Integer Linear Programming (ILP), as well as different heuristic approaches that are based on the principle of ILP relaxation. In contrast to our work, Genez et al. take into account temporally movable and inter-dependent jobs. However, the authors only consider a restricted set of resource types and do not regard fixed-length leasing periods of VMs.

Byun et al. [2] have presented a system for the deployment of workflows within grids and clouds. The authors introduce a heuristic approach for the minimization of leasing costs; they do not provide an exact approach though. Byun et al. consider job dependencies and flexible start times, which are not regarded in our work. However, while they consider fixed-length leasing periods, the authors assume identical leasing instants for all compute hosts, whereas our work permits independent leasing instants for the individual VMs.

In summary, to the best of our knowledge, this work is the first to address the distribution of workloads, i. e., set of jobs, onto VMs under the conjoint consideration of fixed-length leasing periods, as well as fixed and variable price components. Due to the prevalence of these characteristics in actual IaaS cloud offers, our approach permits to compute accurate, i. e., truly cost-optimal, distribution schemes. However, in contrast to the large body of research on workflow deployment, our work does neither explicitly regard dependencies between jobs nor the potential to temporally shift jobs. The consideration of these aspects will be part of our future work.

6 Summary and Outlook

Cloud computing has made a large pool of computing capacity available at comparatively low prices. This permits end users to execute workloads using leased infrastructure rather than costly dedicated hardware. However, given the differing pricing schemes for IaaS offers, which commonly also feature fixed-length leasing periods, the cost-efficient distribution of such workloads among VMs is a challenging task.

In the work at hand, we have addressed this *Cloud-oriented Workload Distribution Problem* (CWDP). As a first major contribution, we have formally described the problem in the form of an mathematical optimization model, which can serve for the computation of exact solutions to the CWDP. As a second major contribution, we have introduced a heuristic optimization approach.

We have quantitatively evaluated both optimization approaches with respect to computation time and solution quality, i. e., resulting total leasing cost. We found that the exact solution approach is hardly applicable to workloads involving more than 20 jobs due to its computational complexity. In comparison, the proposed heuristic allows reductions in computation time of more than 97.5%, which renders it potentially suitable for solving large-scale CWDPs. However, the heuristic solutions also lead to substantial increases in total leasing costs of up to 13% in our experimental evaluation, as compared to an optimal distribution scheme.

Thus, the primary goal of our future work consists in the development of heuristic improvement procedures that are specifically tailored to the CWDP. In addition, we aim to substantially extend the evaluation of the proposed approaches through the consideration of additional variables. Lastly, as it has been outlined in Section 5, we plan to consider characteristics that are common in the area of workflow deployment, such as flexible job initiation times.

Acknowledgments This work has partly been sponsored by E-Finance Lab e. V., Frankfurt am Main, Germany (`http://www.efinancelab.de`).

References

1. Buyya, R., Yeo, C., Venugopal, S., Broberg, J., Brandic, I.: Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. Future Generation Computer Systems 25(6), 599–616 (2009)
2. Byun, E., Kee, Y., Kim, J., Maeng, S.: Cost Optimized Povisioning of Elastic Resources for Application Workflows. Future Generation Computer Systems 27, 1011–1026 (2011)
3. Domschke, W., Drexl, A.: Einführung in Operations Research. Springer, 6th edn. (2004), in German.
4. Genez, T., Bittencourt, L., Madeira, E.: Workflow Scheduling for SaaS/PaaS Cloud Providers Considering two SLA Levels. In: 2012 IEEE Network Operations and Management Symposium. pp. 906–912 (2012)
5. Hillier, F., Lieberman, G.: Introduction to Operations Research. McGraw-Hill, 8th edn. (2005)
6. Lampe, U., Mayer, T., Hiemer, J., Schuller, D., Steinmetz, R.: Enabling Cost-Efficient Software Service Distribution in Infrastructure Clouds at Run Time. In: 2011 IEEE Int. Conf. on Service Oriented Computing & Applications. pp. 82–89 (2011)
7. Li, W., Tordsson, J., Elmroth, E.: Virtual Machine Placement for Predictable and Time-Constrained Peak Loads. In: 8th Int. Workshop on Economics of Grids, Clouds, Systems, and Services. pp. 120–134 (2011)