2015 IEEE 8th International Conference on Cloud Computing

Cost-Efficient Scheduling of Elastic Processes in Hybrid Clouds

Philipp Hoenisch*, Christoph Hochreiner*, Dieter Schuller‡, Stefan Schulte*, Jan Mendling†, Schahram Dustdar*
*Vienna University of Technology, Austria
Email: {p.hoenisch, c.hochreiner, s.schulte, dustdar}@infosys.tuwien.ac.at

‡Technische Universität Darmstadt, Germany
Email: schuller@kom.tu-darmstadt.de

†Vienna University of Economics and Business, Austria
Email: jan.mendling@wu.ac.at

***Abstract*—Cloud computing is becoming increasingly important for executing business processes. This development contributes to a novel class of Business Process Management Systems, called eBPMS, that inherit elasticity from cloud computing.**

The aim of eBPMS is to improve the efficiency of process enactment, in particular regarding scalability and cost-efficiency. However, there is hardly any research that investigates scheduling for eBPMS so far. Against this background, we design an elastic scheduling approach for eBPMS and a corresponding formal problem definition in order to evaluate its data transfer capabilities – this is especially important for hybrid cloud environments. Through extensive evaluations, we are able to show that our approach reduces the total cost by a considerable share.

***Keywords*—Cloud Computing, Hybrid Clouds, Elastic Processes, Optimization, Scheduling, Business Process Management**

I. Introduction

Business Process Management (BPM) is a well-established concept to provide value-added services to customers [1]. Business processes are composed of software-based or human-provided services and span from simple sequences comprising a few process steps up to complex structures which involve splits, e.g., AND, XOR or loops over several dozens of process steps [2]. In order to execute such business processes, Business Process Management Systems (BPMS) are required. BPMS support the complete BPM lifecycle, including process enactment [3]: For this, a BPMS receives new process instance requests and assigns the process steps to designated services and also manages the resource allocation for these services.

Process requests may occur in regular intervals or in an ad hoc manner. Due to these changing process request patterns, a BPMS is exposed to continuously changing resource requirements to enact the requested processes. This is especially the case in extensive and volatile process landscapes where software services implement the corresponding process steps. Rapidly changing resource requirements can lead to either over- or under-provisioning scenarios if the BPMS is working on a fixed amount of resources [4]: In an over-provisioned system environment, the BPMS is provided with too many resources which are not fully used during off-peak times and hence, this over-provisioning leads to a waste of resources and money. In contrast, in an under-provisioned system environment, the BPMS is negatively affected by a lack of resources. Thus, the system will not be able to handle peak loads. Consequently, process executions may result in a low Quality of Service (QoS) and Service Level Agreements (SLAs) may be violated which may result in penalty cost [5].

The emergence of cloud computing offers a promising solution to address these issues: Instead of constantly supplying potentially over- or undersized fixed resources, the resources are only leased when required [6]. Thus, a BPMS is able to provide *elastic* processes, i.e., business processes enacted in an elastic manner using cloud resources. Elastic BPMS (eBPMS) inherit 3 distinctive features of cloud computing: (i) leasing and releasing of computational resources for process execution in an *on-demand, utility-like fashion*, (ii) *rapid elasticity* through scaling resources up or down dynamically, and (iii) *metered service*, allowing pay-per-use [6], [7].

Nevertheless, up to now, there are only very few approaches which consider the usage of cloud resources for process enactment [4]. Especially, dedicated approaches towards process scheduling are missing, even though the introduction of dynamic resource provisioning for BPMS increases the complexity of process scheduling substantially [8], [9]. There are, to the best of our knowledge, so far no scheduling approaches for elastic processes which take into account data transfer aspects. This may be explained by the fact that scheduling approaches for elastic processes usually address a private cloud setting, where data transfer is not considered to be a crucial factor [10]. However, outsourcing of particular services from a private cloud to a public cloud provider is often named as an important business area [11]. Also, the usage of interconnected cloud computing environments, i.e., the combination of cloud resources due to benefits like avoidance of vendor lock-ins, geographical distribution (low latency), or wider resource availability, has recently gained much popularity [12].

This work contributes to the emerging research area of elastic processes by considering data transfer aspects for scheduling and resource provisioning. Furthermore, we facilitate the usage of hybrid clouds for elastic process enactment. For this, we formulate data transfer-aware process scheduling in hybrid clouds as an optimization problem. The optimal solution to the optimization model is then computed by applying Mixed Integer Linear Programming (MILP) techniques. We implement our solution, evaluate it extensively and demonstrate the cost savings that can be obtained if data transfer aspects are explicitly considered for elastic process scheduling.

2159-6190/15 $31.00 © 2015 IEEE
DOI 10.1109/CLOUD.2015.13

The remainder of this paper is structured as follows: We start with a motivational scenario in Section II and state different prerequisites for our approach in Section III. Afterwards, we present our scheduling approach in Section IV. In Section V, our solution is evaluated using a testbed-driven approach, Section VI discusses the related work and Section VII concludes the paper and provides an outlook on our future work.

II. Motivational Scenario

In this section, we provide a motivational scenario to illustrate our work. We consider a scenario from the manufacturing domain [13] since the manufacturing industry is currently undergoing a massive transition towards more effective and interconnected factories [14]. The most important objective is to obtain cost-efficient production processes, which includes an optimal process enactment of the corresponding business processes.

We consider a large factory, which is structured by numerous business processes consisting of different kinds of software-based services. These services range from short, but resource intensive operations, like model rendering or image processing for quality monitoring purposes, to long-running analytic processes. Fig. 1 represents one exemplary business process which is composed of the software-based services S1 to S6. The services deal with images as well as detailed models, which have to be transferred from one service to the next one. This transfer requires time that has to be considered by the BPMS. The factory maintains a fixed set of computational resources, represented by the private cloud in Fig. 1, which is sufficient to instantiate all software-based services in off-peak times. In peak times, when there are numerous parallel process enactments, it is required to lease additional resources from a public cloud to cope with the resource requirements for the services. Fig. 1 represents a snapshot regarding the resource allocation for a peak time scenario, where services S1, S2, S5 and S6 are executed on the fixed resources and services S3 and S4 have to be executed on the public cloud since there are too little resources on the private cloud. The data transfer capabilities within one cloud are very good, so that data transfer of images only requires a short period of time. In contrast, the transfer capabilities between the private cloud and the public cloud are limited, since they are routed over the Internet. These inter-cloud data transfers also issue data transfer cost [10]. They are marked with the exclamation marks in Fig. 1 and the goal is to avoid these costly data transfers during process enactment. Since the in-time completion of the process executions is vital for a cost-efficient production process, each process execution is assigned with a deadline and when a process execution does not meet this deadline, penalty cost accrue.

The magnitude of the process landscape and the strict process execution deadlines can be also found in other domains, like the financial industry [15]. Therefore, the work at hand can also be applied to other domains.

III. Preliminaries

Based on the motivational scenario, we define some preliminaries to realize a data transfer aware scheduling and resource provisioning approach in hybrid clouds.

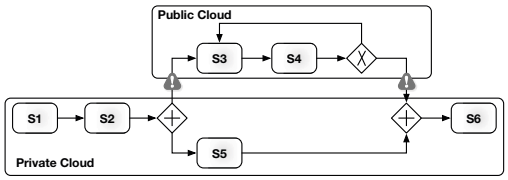

Fig. 1: Motivational Scenario

Our work extends the Service Instance Placement Problem (SIPP) approach, which was originally conceptualized for a private cloud setting [16], and so far neither considers hybrid clouds nor any data transfer aspects. The SIPP approach provides a MILP-based solution to a multi-objective scheduling and resource provisioning problem, which is described in detail in Section IV.

The process landscape is made up from *process instances*, which can be requested by clients by instantiating *process models*. Process models are composed of single *process steps*, which are executed by invoking a *service instance* of a particular *service type* that is deployed on a Virtual Machine (VM). Process models include sequences as well as more complex process patterns, i.e., AND, XOR or loops [2].

VMs are leased and released on demand; for practical reasons they only run one service type at the same time. Nevertheless, VMs are capable of processing multiple *service invocations* in parallel. Further, it is possible to instantiate the same service type on more than one VM. Every process request is accompanied with a SLA that states the deadline for the process instance and therefore represents an elastic QoS constraint for the SIPP optimization model [6]. If a process instance does not meet its predefined deadline, penalty cost accrue which increase the total process enactment cost.

As already stated in the motivational scenario, we consider a hybrid cloud that is composed of a private cloud and a public cloud hosted in two geographically different locations. The eBPMS is deployed in the private cloud and covers both the functionalities of a BPMS and of a cloud controller. Although both, the private and the public cloud, are capable of hosting all service types, we assume that the private cloud should be utilized first as its running cost, e.g., energy cost or maintenance cost, are cheaper compared to the leasing cost for the VMs in the public cloud. Therefore, the natural goal of the optimization model is to achieve a high resource utilization of the private cloud before leasing VMs from the public cloud. In general, we only consider inter-cloud data transfer in terms of data transfer cost, as intra-cloud data transfers are free of charge [10]. However, intra-cloud data transfer also takes some time, which has to be considered. Further, we assume that every service instance maintains its own data repository and that data is only transferred on a process step to process step communication basis.

IV. Process Scheduling

Based on the discussed preliminaries, we are now able to define our elastic process scheduling approach. For this, we first briefly present the SIPP [16]. Afterwards, we extend the scheduling problem to also support hybrid clouds.

A. The Service Instance Placement Problem

To execute elastic processes, software services are deployed on cloud-based computational resources, i.e., VMs. As already mentioned in Section I, companies often suffer from the provisioning problem, i.e., to lease enough resources to handle peak loads and prevent over-provisioning during off-peak times. Therefore, the challenge is to lease as many resources as needed while satisfying required QoS levels.

In order to satisfy these aspects, we define a system model and an optimization model which is aiming at minimizing the total *leasing cost* for computational resources, potential *penalty cost* and through our extension, *data transfer cost*. Besides of minimizing the cost, the most important outcome of the optimization problem is an assignment of service invocations to computational resources, i.e., VMs, in an optimal manner. The SIPP, including all its constraints, variable definitions and utility functions is described in a detailed way in [16], however in the remaining section we will provide a short summary of the system as well as the optimization model.

1) System Model: In order to explain the SIPP's underlying optimization problem (see Equation 1) we first have to define the system model. At this point, it has to be mentioned that the optimization problem refers to a certain time period τ_t, and in order to consider time-dependent variables we make use of the parameter t as index. This allows us to trigger the generation of a scheduling and resource provisioning plan by SIPP at an arbitrary point of time, e.g., in the case information gets outdated or new information is available, like additional process instances have been requested, resources are leased or released, or an unexpected error, like a VM connection failure, occurred. We consider multiple process models, and therefore state the set of process models with P where $p \in P = \{1, ..., p^{\#}\}$ indicates one process model. In a certain time period t, a set of process instances are considered for execution (I_p), where $i_p \in I_p = \{1, ..., i_p^{\#}\}$ refers to one certain process instance.

Considering a particular process instance does not necessarily result in invoking the corresponding service instances, since the SIPP tries to minimize the total cost, i.e., the sum of VM leasing cost and penalty cost. Therefore, at the beginning of time period t, it might not yet be required to start the execution of a process instance, as its deadline is far enough in the future and the execution of the process instance can be delayed to a later optimization period.

Service instances which have to be executed for a particular process instance i_p are defined as the set I_{i_p}, where $j_{i_p} \in J_{i_p} = \{1, ..., j_{i_p}^{\#}\}$ refers to a certain service instance. There are 2 special type of steps: those which have already been scheduled in a previous optimization period (defined by ${j_{i_p}}^{run} \in J_{i_p}$) and those which have to be scheduled in the next optimization period (defined as $j_{i_p}^{*} \in J_{i_p}$).

The optimization model is able to support different types of VMs, which vary in their configuration and their cost per Billing Time Unit (BTU). The set of VM types is indicated by V where $v \in V = \{1, ..., v^{\#}\}$ refers to a certain VM type. Each VM type has a specific resource supply, e.g., available CPU power and RAM, indicated by s_v^C and s_v^R. Analogously, the resource demand of a certain service j_{i_p} is defined by $r^C_{(j_{i_p},v)}$ (for CPU) and $r^R_{(j_{i_p},v)}$ (for RAM). A specific VM is defined by $k_v \in K_v = \{1, ..., k_v^{\#}\}$. The leasing of a VM of type v results in cost c_v. The available resources of one VM regarding CPU and RAM are referred to as $f^C_{k_v}$ and $f^R_{k_v}$ which define whether the VM can serve a service instance or not.

B. Optimization Model

$$
\begin{aligned}
\min \; & \sum_{v \in V} c_v \cdot \gamma_{(v,t)} \\
& + \sum_{p \in P} \sum_{i_p \in I_p} c^p_{i_p} \cdot e^p_{i_p} \\
& + \sum_{v \in V} \sum_{k_v \in K_v} \left(\omega^C_f \cdot f^C_{k_v} + \omega^R_f \cdot f^R_{k_v}\right) \\
& - \sum_{p \in P} \sum_{i_p \in I_p} \sum_{j_{i_p} \in J^*_{i_p}} \frac{1}{DL_{i_p} - \tau_t} x_{(j_{i_p},k_v,t)} \\
& + C_D
\end{aligned} \tag{1}
$$

Equation 1 shows the main objective of the SIPP model. The objective comprises 5 terms. The first 4 terms form the original SIPP and are presented in the following. The fifth term C_D is an extension to the SIPP, i.e., this term is dedicated to enable and optimize hybrid cloud deployment of service instances and their invocations, which is explained in detail in Section IV-C.

The first term in Equation 1 computes the total cost accruing due to leasing $\gamma_{(v,t)}$ VMs of type v in a certain leasing period t. The second term computes the cost which may accrue if deadlines are violated, i.e., penalty cost which have to be paid for delayed process instances. For computing these penalties we apply a linear penalty function as described in [5]. This means that each process instance (or more precisely: the SLA of this process instance) includes defined penalty cost $c^p_{i_p}$ which are due if the process instance gets delayed. This value is multiplied with the actual period of time $e^p_{i_p}$ for which the process instance is delayed. The third term in the equation is used to consider the cost of *wasted* resources, i.e., the sum of free resource capacities $f^C_{k_v}$ and $f^R_{k_v}$. The fourth term is used to compute the urgency of process instances, i.e., we subtract the current period in time τ_t from the deadlines DL_{i_p} and compute the corresponding reciprocal value. This way, a process instance having a closer deadline is assigned with a higher priority since the value $\frac{1}{DL_{i_p} - \tau_t}$ gets larger.

C. Scheduling under Consideration of Hybrid Clouds

Usually, private clouds provide a fixed amount of computational resources and therefore suffer from over- and under-provisioning. Having this problem in mind, many companies choose to extend their private cloud with a public cloud to lease theoretically unlimited additional resources to deploy resource-intensive service types [11], [17].

The proposed optimization model from Section IV-B is able to optimize the scheduling and placement of service invocations on computational resources provided by a single cloud (independent from the fact whether it is either a private or a public cloud). However, the support of hybrid clouds raises additional challenges especially in terms of data transfer, i.e.,

the duration it takes to transfer data and the cost which accrue when a large amount of data is moved from or into a cloud. In order to address these challenges, we extended the original SIPP model in Equation 1 with the fifth term C_D. This, and additional constraints and utility functions (Equations 2-6) are explained in the following.

1) Data Transfer Cost: First we consider the data transfer cost, which are represented by the new variable C_D (see Equation 4). This variable represents all data transfer cost which are issued by the inter-cloud data transmission between currently running respectively already finished process steps and future process steps that are currently considered in the scheduling plan.

To distinguish between private cloud VMs and public cloud VMs, we extend the concept of the VM variable K_v: We introduce 2 subtypes of this variable, $K_{v_{priv}}$ which represents VMs in the private cloud and $K_{v_{pub}}$ which represents VMs in the public cloud. Since the original system model of SIPP does not consider any data transfer between 2 sequenced process steps, we introduce the additional variable $j_{i_p}^{past}$. This variable represents all process steps which are currently running or have already been completed to determine whether 2 adjoining process steps issue any data transfer cost. 2 successive process steps can follow 4 different deployment scenarios: In the first 2 scenarios both process steps are either deployed in the private cloud or in the public cloud. As intra-cloud data transfer is free of charge, no data transfer cost will accrue within this scenario. In contrast to that, in the 2 remaining deployment scenarios, the predecessor step can either be deployed in the private cloud and the successor step in the public cloud or vice versa. Within this scenario, data transfer cost will accrue and have to be considered.

The actual deployment situation is evaluated using the utility function $l(j_{i_p}^{past}, j_{i_p}^{*})$ (see Equation 2). In case a scheduling plan consists of a deployment spanning across private and public clouds, we assign data transfer cost based on the output result size of the first task, since hosting providers usually only charge for outgoing data transfer (see Equation 3). In any other case, we assign no data transfer cost. These individual transfer cost $C_{D_{j_{i_p}}}^{\#}$ are accumulated by an additional constraint (see Equation 4) to be considered for the overall optimization objective (see Equation 1) as C_D.

$$l(j_{i_p}^{past}, j_{i_p}^{*}) = \begin{cases} 0, & \text{if } k_1, k_2 \in K_{v_{priv}}, x_{(j_{i_p}^{past}, k_1, t)}, \\ & x_{(j_{i_p}^{*}, k_2, t)} \\ 0, & \text{if } k_1, k_2 \in K_{v_{pub}}, x_{(j_{i_p}^{past}, k_1, t)}, \\ & x_{(j_{i_p}^{*}, k_2, t)} \\ 1, & \text{else} \end{cases} \quad (2)$$

$$C_{D_{j_{i_p}}}^{\#} = \begin{cases} C_{D_{j_{i_p}}} & \text{, if } l(j_{i_p}^{past}, j_{i_p}^{*}) = 1 \\ 0 & \text{, else} \end{cases} \quad (3)$$

$$\sum_{v \in V} \sum_{k \in K_v} \sum_{j_{i_p} \in J_{i_p}} x_{(j_{i_p}, k_v, t)} * C_{D_{j_{i_p}}}^{\#} \leq C_D \quad (4)$$

2) Collocation of Service Invocations: The amount of accruing data transfer cost depends on the service type in combination with the deployment location, i.e., whether it is deployed in the public or private cloud. Using this information, the collocation of successive service invocations is realized based on the data transfer cost constraints in an implicit manner, as the objective function aims at minimizing the overall cost.

3) Selective Usage of Public Resources: To implement the conditional usage of the public cloud, we introduce the utility function u (see Equation 5) and the functionality to assign the future leasing cost for the public VMs based on the current usage of the private cloud. The utility function u sums up the cores of all currently leased private VMs. In order to have a VM-independent resource usage calculation function, the usage calculation is carried out on the basis of VM cores. As long as the usage of the private cloud is below 70%, the leasing cost of the public VMs are multiplied with 100 to make them a non-desirable option for the scheduling plan (see Equation 6). The concrete figure of 70% was chosen based to achieve a high resource utilization for the private cloud, while maintaining necessary reserves in case of VM failures to achieve a data transfer optimized scheduling plan, even if some VMs have to be replaced [18].

$$u = \sum_{k_v \in K_{v_i}} core(g(k_{v_{priv}}, t)) \quad (5)$$

$$c_{v_{ex}} = \begin{cases} c_{v_{ex}} & \text{, if } u > \left(\sum_{k_v \in K_{v_i}} core(k_{v_{priv}}) \right) * 0.7 \\ c_{v_{ex}} * 100 & \text{, else} \end{cases} \quad (6)$$

This constraint does not completely rule out the usage of VMs provided by the public cloud, since the scheduling plan may require a specific type of a VM, which is currently not available in the private cloud and an expensive VM may be a better option than paying penalty cost. Nevertheless, since the cost are higher than for the private resources, the selection of such an expensive VM is unlikely. In general, we assign leasing cost for the private cloud as well as the public cloud. The cost structure for the private cloud enables us to implement a fine-grained selection of different VM types to use the private cloud as efficient as possible.

V. Evaluation

For evaluating our scheduling and provisioning approach we made use of the Vienna Platform for Elastic Processes (ViePEP) [19]. As ViePEP was originally designed for a single private cloud environment, we implemented an extension allowing to lease VMs from an Openstack-based cloud[1] as well as from Amazon's EC2[2]. While the private cloud is restricted in terms of computational resources, the public cloud provides virtually unlimited resources. The BTU for both clouds is set to 5 minutes.

[1] http://www.openstack.org/software/folsom/
[2] http://aws.amazon.com/ec2/

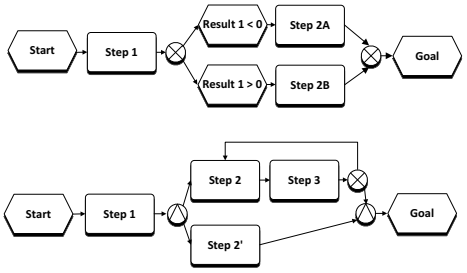

Fig. 2: Process No. 5 (top), Process No. 9 (bottom)

TABLE I: Service Types

Service Type No.	CPU Load in % (μ_{cpu})	Makespan in sec. ($\mu_{makespan}$)	Outgoing data cost
1	50	30	2
2	75	80	4
3	75	120	6
4	100	20	2
5	120	100	4
6	125	30	6
7	150	40	2
8	175	20	4
9	250	60	6
10	310	30	2

On the one hand ViePEP operates on a Platform as a Service (PaaS) level and accepts new process instance requests and on the other hand it operates on the Infrastructure as a Service (IaaS) level to lease and release computational resources in order to enact the requested process instances. Further, ViePEP makes use of CPLEX[3] for solving the SIPP model. The result is a full scheduling and resource allocation plan, i.e., a scheduling plan telling on which VM instance a particular service instance has to be invoked in order to fulfill a certain process instance.

In the following subsections, we first present the evaluation setup in Section V-A. Afterwards, we present the quantitative evaluation including a discussion in Section V-B.

A. Evaluation Setup

1) Process Models: To perform a realistic evaluation we selected 10 representative process models from the SAP reference model [20]. This reference model has been evaluated and used for many scientific papers, e.g., [21] and provides a solid basis for our evaluation. The selected process models are composed from different process patterns and different levels of complexity, i.e., the process models may be composed of AND-splits, XOR-splits or loops. The former two types consist of a split (AND, XOR) and an appropriate merge pattern, which either is blocking (AND) or simply continues the process execution, as soon as 1 optional process step is completed (XOR). Figure 2 shows two sample process models, namely process model 5 which contains an XOR-split, and model 9, which contains an AND-split with a nested loop in one of the AND-paths. The remaining process models range from a simple sequential process with 3 process steps up to complex ones consisting of 20 process steps that are connected with complex process patterns.

2) Test bed: Although real world processes like those in the SAP reference model are composed of software-based and human-provided services, our work focuses on software-based services. For these software-based services, we simulate 10 software service types with different resource requirements and makespans. Table I provides a detailed overview about the 10 simulated service types. The table states the required mean CPU load in percent, the mean makespan in seconds, and the size of outgoing data in cost units. The services' CPU load describes the required amount of CPU in percent on a single core VM. If a service is deployed on a multi-core VM, the load is equally distributed among all of them, i.e., we assume that each service is fully parallelizable. This leads to the fact that only the first 3 services of Table I can be deployed on a single core VM. All other services require at least a dual core VM or an even larger one. Notably, with an increasing number of CPU cores, the makespan remains the same. For generating the CPU load, we use the lookbusy load generator[4]. This is a configurable tool to generate CPU load on a VM for a given time span and enables us to simulate the resource consumption. The values μ_{cpu} and $\mu_{makespan}$ provided in Table I represent mean values of 2 general normal distributions where we assume $\sigma_1 = \mu_{cpu}/10$ respectively $\sigma_2 = \mu_{makespan}/10$. ViePEP is able to lease 7 different VM types: 4 VM types from the private Openstack-based cloud, and 3 VM types from Amazon's EC2 public cloud.

3) Applied SLAs: Each process instance is applied with a SLA defining its deadline, i.e., the latest point of time when the process instance has to be finished. To evaluate the applicability of our approach, we choose two different SLA scenarios. First, we apply a *strict* scenario, i.e., the deadline is defined by 1.5 times the process model's average makespan. Second, we apply a *lenient* scenario, i.e., the deadline is defined by 2.5 times the process model's average makespan. The 2 values were chosen due to the fact that VMs require some time for the deployment of the services and individual VMs may fail, so that ViePEP has to lease replacement VMs which affects the overall process execution negatively.

4) Request Patterns: We apply 2 different request patterns: First, we follow a *constant* request pattern, for which we select 7 process models every 120 seconds. In order to vary the process models, we create different instances based on the round-robin principle, i.e., in the first round process models 1 to 7 are requested, in the next round 8 to 10 and 1 to 4, and so on. This is repeated until a total of 70 process instances are requested in a timeframe of about 20 minutes.

The second arrival pattern follows a *pyramid*-like function, which is represented by Equation 7. Variable a represents the amount of process instance requests at the single points in time n every 120 seconds. The actual process selection aims for a realistic distribution among common and rare process models. To achieve this variable distribution, we start for every new request batch with process number 1 and increment the process number to match the amount of process requests. The second request pattern issues in total 118 process instances within a timeframe of 48 minutes.

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

[4]http://devin.com/lookbusy/

$$f(n) = \begin{cases} a = n & \text{if } 0 \leq n \leq 5 \\ a = n/2 & \text{if } n = 6 \\ a = 1 & \text{if } 7 \leq n \leq 10 \\ a = n - 10 & \text{if } 11 \leq n \leq 19 \\ a = 10 & \text{if } 20 \leq n \leq 24 \end{cases} \quad (7)$$

5) Baseline for the Evaluation: We compare our evaluation against a *baseline* which follows a basic scheduling and resources provisioning strategy [22]: As soon as a service instance needs to be invoked and no VM hosting a specific service is available, a quad core VM is leased and the corresponding service type is deployed. In addition, to prevent under-provisioning, an additional quad core VM will be leased and the needed service type will be deployed, if a VM's resource utilization level reaches 80%. If a VM's resource load is below 20%, the VM will be released as soon as all service invocations on this VM have been finished. Within the baseline, VMs are equally leased from the public as well as from the private cloud without considering any data transfer aspects. Nevertheless, all restrictions applied to the SIPP model, like only 1 service type per VM, QoS constraints, i.e., the given deadline, and the issued cost are also applied for the baseline approach.

6) Metrics: To assess the performance of our scheduling approach, we apply different metrics: First, the *Total Makespan* is measured, i.e., the overall time span from the first process instance request until the successful termination of the last process step of the last process instance. Second, the *Total Cost* are computed. They are composed from *Private Leasing Cost* representing the leasing cost for the private cloud, *Public Leasing Cost* which represent the leasing cost for the public cloud and *Data Transfer Cost* which are charged for transferring data across the different clouds. In addition, the Total Cost also include *Penalty Cost* which accrue if a process instance is delayed. For that, we apply according to the delay, 1 cost unit penalty for every 10% of the overall makespan of the process model. We further assess the *SLA Adherence*, i.e., the percentage of process executions which are compliant with their SLA. Regarding the leasing cost, we assume that it is cheaper to lease a dual core VM than two single core VMs, i.e., we follow a cost model with decreasing marginal cost.

To receive significant data, we perform 3 iterations for each process request pattern combined with the 2 different SLA levels resulting in a total of 12 runs for our approach as well as 12 for the baseline.

B. Results and Discussion

The following section presents and discusses the results of our evaluation by listing all key metrics in Table II. Notably, the numbers in this table, as well as the numbers below in the discussion represent the mean value over the 3 runs. The standard deviation σ is also presented in the table.

First we discuss the constant request pattern for both SLA scenarios: The most prominent difference between the baseline and our extended SIPP approach are the higher total cost for the baseline approach. The baseline approach for the lenient SLA scenario issues 1.84 times the total cost of the extended SIPP approach and the strict SLA scenario even requires 2.59 times the total cost for the same amount of process requests. As it can be seen in the table, the leasing cost represent the major part of the overall cost. Overall, in our approach, less VMs from the public cloud were leased leading to significant smaller leasing cost. These results show that the selective usage of public resources works as intended and the extended SIPP only leases public resources when they are required to comply with the SLA. This has a large impact on the overall data transfer cost. The baseline issued about 14 times as much data transfer cost which represent almost a third of its total cost, i.e., 26% for the strict SLA scenario and 30% for the lenient one. Compared to our approach, where data transfer cost amounts to only 4% of the total cost for the strict scenario (and none for the lenient one). While observing the SLA adherence, we see that hardly any SLAs are violated in the constant scenario. The fact that we experienced no SLA violations in the strict scenario for our approach, can be reduced to the fact that the overall deadlines are *close* which results in a higher risk of delay compared to the lenient scenario. As penalty cost are an important factor, closer deadlines *force* earlier scheduling. If a VM is already leased, the SIPP model aims at reducing *wasting* resources, hence, future process instances are preponed.

Next to the SLA adherence it remains to discuss the overall makespan: In general, the strict scenario finishes faster than the lenient one, and further, the baseline finishes faster than our approach. This can be reduced to the fact that the baseline leases more VMs, hence, has more resources available and is able to perform more service invocations in parallel. In contrast, the extended SIPP only leases additional resources when they are cost-efficient in respect to the penalty cost model. However, although the baseline was faster than the extended SIPP, it also results in much higher cost ($\sim$2.5 times).

Second, we discuss the pyramid request scenario where 48 additional process instances were requested. Due to the nature of this scenario, both approaches had to lease in total more resources than in the constant scenario leading to higher leasing cost of both, public VMs and private VMs. Interesting to see is the ratio between data transfer cost and leasing cost: For our approach, this ratio amounts to 8% for the strict SLA and 5% for the lenient one. For the baseline it is 36% for the strict SLA scenario and 59% for the lenient one. This means, that our approach preferred to schedule service invocations within the same cloud over switching between the public and private cloud. As the baseline does not consider any data transfer cost for its scheduling plan, it results in much higher cost. Due to the used penalty cost model, the overall SLA adherence is in general lower (57%-86%) compared to the constant request scenario (90%-100%) as penalty cost are comparably cheaper than leasing additional resources. The reason that we experienced more SLA violations using the extended SIPP model in the pyramid request scenario than for the constant request scenario, can be attributed to the fact that in total more process instances are processed.

The evaluations have shown that the usage of our approach for creating a scheduling plan which considers leasing cost, penalty cost, and data transfer cost significantly decreases the overall cost comparing to an ad hoc approach. The numbers in Table II reveal that the extended SIPP approach was able to reduce the overall cost by an average of 53% for the constant requests and an average of 45% for the pyramid requests.

TABLE II: Evaluation Results

	Constant				Pyramid			
	Extended SIPP		Baseline		Extended SIPP		Baseline	
SLA Level	Strict	Lenient	Strict	Lenient	Strict	Lenient	Strict	Lenient
Number of Process Requests	70				118			
Interval between two Process Requests	120 seconds							
Number of Parallel Process Requests	$y = 7$				$f(n)$ (see Equation 7)			
Total Makespan in Minutes (Standard Deviation)	26.33 (σ=2.30)	33.00 (σ=3.00)	23.67 (σ=1.15)	32.67 (σ=7.51)	62.00 (σ=2.00)	67.33 (σ=0.59)	58.67 (σ=0.59)	61.67 (σ=0.59)
SLA Adherence in % (Standard Deviation)	100.00 (σ=0.00)	90.00 (σ=7.95)	94.76 (σ=6.60)	100.00 (σ=0.00)	57.91 (σ=4.67)	76.27 (σ=5.29)	60.92 (σ=5.06)	86.72 (σ=3.42)
Penalty Cost (Standard Deviation)	0.00 (σ=0.00)	10.33 (σ=8.02)	15.00 (σ=24.24)	0.00 (σ=0.00)	155.33 (σ=34.43)	58.00 (σ=19.28)	123.33 (σ=15.00)	27.33 (σ=9.87)
Private Leasing Cost (Standard Deviation)	963.33 (σ=77.36)	1284.00 (σ=300.39)	1061.67 (σ=20.20)	385.00 (σ=121.24)	1793.00 (σ=47.70)	1934.67 (σ=199.58)	1458.33 (σ=253.20)	1540.00 (σ=185.20)
Public Leasing Cost (Standard Deviation)	45.00 (σ=0.00)	0.00 (σ=0.00)	945.00 (σ=242.49)	1061.67 (σ=80.83)	420.00 (σ=108.17)	548.33 (σ=137.69)	1796.67 (σ=225.02)	1563.33 (σ=359.21)
Data Transfer Cost (Standard Deviation)	52.00 (σ=13.85)	0.00 (σ=0.00)	724.00 (σ=187.06)	937.33 (σ=244.79)	170.67 (σ=94.77)	136.67 (σ=28.94)	1184.67 (σ=193.67)	1051.33 (σ=204.92)
Total Cost (Standard Deviation)	1060.33 (σ=91.22)	1294.33 (σ=308.39)	2745.67 (σ=474.00)	2384.00 (σ=204.38)	2539.00 (σ=111.53)	2677.67 (σ=216.15)	4563.00 (σ=153.27)	4182.00 (σ=481.38)

VI. Related Work

While scheduling for elastic processes is still an emerging research topic, a small number of corresponding solution approaches have emerged in recent years [4]. Furthermore, there are several resource provisioning strategies for single applications, e.g., [5], [23]. These approaches do not take into account any process perspective. Instead, they focus on ad hoc scheduling and resource provisioning of single applications. In addition, there are a number of scheduling approaches for scientific workflows, e.g., [10], [24].

Resource constraints – the primary driving force behind scheduling for elastic processes – are a relatively neglected topic in process enactment [25]. To the best of our knowledge, there is currently no approach providing optimal scheduling in hybrid clouds while taking into account data transfer aspects. Nevertheless, there are some approaches which are related to our work: Juhnke et al. provide an extension to a BPEL engine which allows to apply cloud-based computational resources to carry out process steps [9]. Notably, the authors take into account data transfer duration and cost. SLAs for single process enactments are not regarded, since the optimization aims at (weighted) pareto-efficient scheduling related to cost and turnaround duration. Also, scheduling is done with regard to one single cloud and for single processes. Bessai et al. provide a similar approach, also aiming at optimization of either cost, turnaround duration or a priority-based pareto-efficient combination thereof [26]. Processes do not share VMs and SLAs for single processes are not regarded. Nevertheless, data transfer duration and cost are taken into account for the optimization. Although the authors take into account different public clouds, they do not consider hybrid ones.

In our former work [16], [27], [28], we have provided the basics for scheduling elastic processes in the cloud, taking into account sharing of services among concurrently running processes and SLAs for single process instances. Optimizations have been done with regard to cost and deadline constraints. However, we have not taken into account data transfer issues and hybrid clouds. Therefore, the work at hand provides a substantial extension of our former work.

Apart from the already discussed scheduling solutions, there are several other approaches to elastic process scheduling which do not take into account data transfer aspects: Wei et al. aim at optimizing overall resource utilization, but deadline-constrained scheduling is not explicitly considered [29]. Euting et al. apply fuzzy theory in order to ensure deadline-aware scheduling [30]. Wu et al. explicitly aim at minimizing the cost for a hybrid cloud-based process landscape [31]. For this, a hierarchical scheduling approach is provided, which first tries to locate services; if services are not available, new service instances are deployed in a local cloud and in a second step, service allocations in a private cloud are optimized.

Taking into account the methodology applied in our contribution, an approach by Cai et al. comes closest to our optimization solution, as the authors also apply MILP [32]. However, scheduling is only done for single process instances and resources are not shared amongst processes.

VII. Conclusion and Future Work

Within this paper we have shown that considering data transfer aspects while creating a scheduling plan for hybrid clouds can heavily reduce the total cost. In order to achieve an optimal scheduling plan we integrated additional constraints into the original SIPP model [16]. Our evaluations have shown that our optimization model is able to schedule service invocations among hybrid cloud resources in a cost-efficient way. It also does not just reduce the leasing cost but also reduces the data transfer cost compared to an ad hoc baseline. Using the extended SIPP approach, we achieved total cost savings of up to 45%-53%, depending on the process requests.

However, our evaluations have also shown that the cost reduction depends on the underlying cost model, e.g., we experienced some SLA violations since it was *cheaper* to accept a delay instead of leasing additional resources. Based on this insight, we plan to use different cost models for penalty cost, data transfer cost and leasing cost in our future work. In addition, up to now we focused on process step to process step communications, i.e., the output of one process step is directly used as input for the next process step. However, real world

use cases often incorporate more complex data patterns, like shared global storage systems. Future evaluations will consider these complex data patters, a larger set of services as well as different cloud providers.

Another important research area for hybrid clouds are privacy restrictions for service types or the processed data. These privacy restrictions could either be issued by the legislation for sensitive data or are based on precautions not to deploy sensitive trade secrets, e.g., algorithms or customer data, on a public cloud. These constraints require an extended set of SLAs which have to be considered.

ACKNOWLEDGMENT

This paper is supported by TU Vienna research funds. This work is partially supported by the Commission of the European Union within the CREMA H2020-RIA project (Grant agreement no. 637066).

REFERENCES

[1] M. Rosemann and J. vom Brocke, "The Six Core Elements of Business Process Management," in *Handbook on Business Process Management 1*. Springer, 2010, pp. 107–122.

[2] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow Patterns," *Distributed and Parallel Databases*, vol. 14, no. 1, pp. 5–51, 2003.

[3] M. Weske, *Business Process Management: Concepts, Languages, Architectures*, 2nd ed. Springer, 2012.

[4] S. Schulte, C. Janiesch, S. Venugopal, I. Weber, and P. Hoenisch, "Elastic Business Process Management: State of the Art and Open Challenges for BPM in the Cloud," *Future Generation Computer Systems*, vol. 46, pp. 36–50, 2015.

[5] P. Leitner, W. Hummer, B. Satzger, C. Inzinger, and S. Dustdar, "Cost-Efficient and Application SLA-Aware Client Side Request Scheduling in an Infrastructure-as-a-Service Cloud," in *5th International Conference on Cloud Computing (CLOUD 2012)*. IEEE, 2012, pp. 213–220.

[6] S. Dustdar, Y. Guo, B. Satzger, and H. L. Truong, "Principles of Elastic Processes," *IEEE Internet Computing*, vol. 15, no. 5, pp. 66–71, 2011.

[7] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," *Communications of the ACM*, vol. 53, pp. 50–58, 2010.

[8] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic Placement of Virtual Machines for Managing SLA Violations," in *10th IFIP/IEEE International Symposium on Integrated Network Management (IM'07)*. IEEE, 2007, pp. 119–128.

[9] E. Juhnke, T. Dörnemann, D. Bock, and B. Freisleben, "Multi-objective Scheduling of BPEL Workflows in Geographically Distributed Clouds," in *4th International Conference on Cloud Computing (CLOUD 2011)*. IEEE, 2011, pp. 412–419.

[10] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, "Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 158–169, 2013.

[11] R. van den Bossche, K. Vanmechelen, and J. Broeckhove, "Online cost efficient scheduling of deadline-constrained workloads on hybrid clouds," *Future Generation Computing Systems*, vol. 29, no. 4, pp. 973–985, 2013.

[12] A. N. Toosi, R. N. Calheiros, and R. Buyya, "Interconnected Cloud Computing Environments: Challenges, Taxonomy and Survey," *ACM Computing Surveys*, vol. 47, no. 1, 2014.

[13] S. Schulte, P. Hoenisch, C. Hochreiner, S. Dustdar, M. Klusch, and D. Schuller, "Towards Process Support for Cloud Manufacturing," in *18th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2014)*. IEEE, 2014, pp. 142–149.

[14] S. Dowlatshahi and Q. Cao, "The relationships among virtual enterprise, information technology, and business performance in agile manufacturing: An industry perspective," *European Journal of Operational Research*, vol. 174, no. 2, pp. 835–860, 2006.

[15] U. Lampe, O. Wenge, A. Müller, and R. Schaarschmidt, "On the Relevance of Security Risks for Cloud Adoption in the Financial Industry," in *19th Americas Conference on Information Systems (AMCIS 2013)*. AIS, 2013.

[16] P. Hoenisch, D. Schuller, S. Schulte, C. Hochreiner, and S. Dustdar, "Optimization of complex elastic processes," *IEEE Transactions on Services Computing*, vol. NN, no. NN, pp. NN–NN, 2015, to appear.

[17] L. F. Bittencourt and E. R. M. Madeira, "HCOC: A Cost Optimization Algorithm for Workflow Scheduling in Hybrid Clouds," *Journal of Internet Services and Applications*, vol. 2, no. 3, pp. 207–227, 2011.

[18] Q. Zheng and B. Veeravalli, "Utilization-based pricing for power management and profit optimization in data centers," *Journal of Parallel and Distributed Computing*, vol. 72, no. 1, pp. 27–34, 2012.

[19] S. Schulte, P. Hoenisch, S. Venugopal, and S. Dustdar, "Introducing the Vienna Platform for Elastic Processes," in *Performance Assessment and Auditing in Service Computing Works. (PAASC 2012) at 10th International Conference on Service Oriented Computing (ICSOC 2012)*, ser. LNCS, vol. 7759. Springer, 2013, pp. 179–190.

[20] T. A. Curran and G. Keller, *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Prentice Hall PTR, Upper Saddle River, 1997.

[21] J. Mendling, H. M. W. Verbeek, B. F. van Dongen, W. M. P. van der Aalst, and G. Neumann, "Detection and prediction of errors in EPCs of the SAP reference model," *Data & Knowledge Engineering*, vol. 64, no. 1, pp. 312–329, 2008.

[22] M. E. Frincu, S. Genaud, and J. Gossa, "On the Efficiency of Several VM Provisioning Strategies for Workflows with Multi-threaded Tasks on Clouds," *Computing*, vol. 96, pp. 1059–1086, 2014.

[23] H. Li and S. Venugopal, "Using Reinforcement Learning for Controlling an Elastic Web Application Hosting Platform," in *8th International Conference on Autonomic Computing (ICAC 2011)*. ACM, 2011, pp. 205–208.

[24] E.-K. Byun, Y.-S. Kee, J.-S. Kim, and S. Maeng, "Cost optimized provisioning of elastic resources for application workflows," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1011–1026, 2011.

[25] Z. Huang, W. M. P. van der Aalst, X. Lu, and H. Duan, "Reinforcement learning based resource allocation in business process management," *Data & Knowledge Engineering*, vol. 70, no. 1, pp. 127–145, 2011.

[26] K. Bessai, S. Youcef, A. Oulamara, C. Godart, and S. Nurcan, "Bi-criteria workflow tasks allocation and scheduling in cloud computing environments," in *5th International Conference on Cloud Computing (CLOUD 2012)*. IEEE, 2012, pp. 638–645.

[27] P. Hoenisch, S. Schulte, S. Dustdar, and S. Venugopal, "Self-Adaptive Resource Allocation for Elastic Process Execution," in *6th International Conference on Cloud Computing (CLOUD 2013)*. IEEE, 2013, pp. 220–227.

[28] S. Schulte, D. Schuller, P. Hoenisch, U. Lampe, R. Steinmetz, and S. Dustdar, "Cost-Driven Optimization of Cloud Resource Allocation for Elastic Processes," *International Journal of Cloud Computing*, vol. 1, no. 2, pp. 1–14, 2013.

[29] Y. Wei and M. B. Blake, "Proactive virtualized resource management for service workflows in the cloud," *Computing*, vol. 96, no. 7, pp. 1–16, 2014.

[30] S. Euting, C. Janiesch, R. Fischer, S. Tai, and I. Weber, "Scalable Business Process Execution in the Cloud," in *International Conference on Cloud Engineering (IC2E 2014)*. IEEE, 2014, pp. 175–184.

[31] Z. Wu, X. Liu, Z. Ni, D. Yuan, and Y. Yang, "A market-oriented hierarchical scheduling strategy in cloud workflow systems," *The Journal of Supercomputing*, vol. 63, no. 1, pp. 256–293, 2013.

[32] Z. Cai, X. Li, and J. N. Gupta, "Critical Path-Based Iterative Heuristic for Workflow Scheduling in Utility and Cloud Computing," in *11th International Conference on Service Oriented Computing (ICSOC 2013)*, ser. LNCS, vol. 8274. Springer, 2013, pp. 207–221.