Optimizing the Distribution of Software Services in Infrastructure Clouds

Ulrich Lampe (Advisor: Prof. Dr.-Ing. Ralf Steinmetz)
Multimedia Communications Lab (KOM) – Technische Universität Darmstadt, Germany
E-Mail: ulrich.lampe@KOM.tu-darmstadt.de

***Abstract*—In the public perception, cloud computing is frequently associated with almost unlimited elasticity and scalability of computing capacity. However, Infrastructure as a Service in the form of virtual machines provides limited supplies of virtual resources, due to restrictions of the underlying physical hardware. At the same time, the execution of Software as a Service instances leads to a specific demand for these resources. Based on this observation, I introduce the *Software Service Distribution Problem*, i.e., the challenge of (cost-)efficiently distributing the execution of software service instances across available cloud infrastructure providers and virtual machine types under resource constraints. I outline my research approach, which aims at the development of optimization algorithms in the context of an integrated *Software Service Distribution Broker*, and report the progress made to date.**

I. Introduction

In recent years, cloud computing has evolved as a novel *Information Technology* (IT) paradigm for the provision of computing capacity in a utility-like fashion [1]. A common promise and perception is that cloud computing natively offers almost unlimited elasticity and scalability of computing capacity.

In fact, this is largely true in the case of *Platform as a Service* (PaaS) offers, such as *Google's App Engine*, which automatically *scale out* according to the requirements of the executed software. However, PaaS requires the adaptation of the deployed software to a proprietary platform [2]. In the case of existing (legacy) software or for many application domains, this restriction can be problematic – in fact, it can be *the* show-stopper for the utilization of PaaS.

In contrast to PaaS, *Infrastructure as a Service* (IaaS) – specifically in the form of *Virtual Machines* (VMs), such as provided by *Amazon's Elastic Compute Cloud* (EC2) – offers a fully customizable environment for software execution. In exchange, the provided VMs only *scale up* to the level of the physical machine hosting them. Thus, an explicit instantiation of VMs is required depending on the current compute load.

Still, if software is to be deployed in the cloud in a platform-independent manner, IaaS is necessarily the model of choice. In this case, however, aforementioned resource restrictions apply. As it will be explained in this work, this situation leads to a novel research challenge concerning the cloud-based provision of software services.

The remainder of this paper is structured as follows: In Section II, a detailed description of the scenario and resulting research problem will be provided. Section III gives an overview of the current state of research. In the following Sections IV and V, I will outline my research objective and corresponding approach. Section VI provides preliminary results. Lastly, Section VII concludes the paper with a brief summary and a description of future work.

II. Research Problem

A. Motivation & Scenario

In my ongoing work, I adapt the three-layered cloud model by Armbrust et al. [3]. In this model, a *SaaS provider* implements software in the form of *Software as a Service* (SaaS). Instances of the resulting software services are requested by *SaaS users* and executed using the leased infrastructure, i.e., VM instances, of various *IaaS providers*. Again, in short, various SaaS users request *Software Service Instances* (SSIs) from a SaaS provider, who in turn utilizes the leased VMs of different IaaS providers for the execution of these SSIs.

A key observation is that each requested SSI can be associated with a specific resource demand, e.g.,

in terms of processor power or memory consumption. At the same time, each of the available VM types provides a certain quantity of these resources. Amazon's EC2, for instance, currently offers eight different VM types with varying resource supplies and hourly operating costs. Thus, the SaaS provider faces the challenge of distributing the requested SSIs across the available VM instances, which constitutes a combined capacitation (number of leased instances of each VM type) and assignment (allocation of SSIs to specific VM instances) problem.

A practical example of aforementioned situation is cloud gaming: Video games are executed on VMs, which transmit the resulting audio/video stream to end users' playback devices and receive control commands from them. The video games, i.e., SSIs, may be executed at different visual quality levels, depending on, e.g., the display resolution of the respective playback device. This results in a specific and, throughout the execution of the game, essentially static resource demand on the VM. Again, at the same time, the VM instance that hosts the game is subject to a restricted supply of these resources.

B. Software Service Distribution Problem

The outlined scenario leads to a novel research problem, which in the following will be referred to as *Software Service Distribution Problem* (SSDP). In its basic form, this problem concerns the SaaS provider and consists in the distribution of a set of requested SSI executions across a set of leased VMs, such that:

1) the resource demands of all SSIs are met by the available resource supplies of the respective VM instances where the SSIs are executed (i.e., the distribution is *effective*)
2) the overall cost of leasing the required VMs is minimized (i.e., the distribution is *efficient*).

C. Problem Extensions

Apart from the restrictions concerning resource types, the SSDP can be subject to multiple other, partially interdependent constraints. In the following, a selected set of aspects that I will consider in my work are outlined.

Quality of Service: SSI executions are often subject to certain Quality of Service (QoS) requirements, e.g., with respect to response time or availability. For that matter, each SSI request could be associated with QoS requirements that have to be matched against the QoS guarantees of the available VM types.

Substitutable Resources: The demanded resource types in SSI executions may be substitutable to some extent (e.g., the use of traffic compression results in lower bandwidth demands, but higher processor utilization). Subsequently, the SSDP should consider alternative resource demands that result from different substitution options and patterns.

Stochastic Resource Demands: Mining historic SSI execution data will usually provide a stochastic distribution of resource demands, rather than deterministic (fixed) values. These distributions may, for instance, be regarded through the quantification of risks that resource constraints are broken in SSI execution.

Environmental Impact: The available VM types may not only differ with respect to their price (*economic* cost), but also with respect to their environmental impact (*ecological* cost). For instance, the availability of renewable energy may reduce greenhouse emissions of a data center that hosts VMs. Thus, the reduction of harmful environmental effects may constitute an additional objective of the SSI distribution process.

Pricing Models: Cloud IaaS providers do not necessarily employ a (linear) pay-per-use pricing model. Accordingly, the service distribution process should consider additional pricing mechanisms, such as auctions or volume-discounted rates.

III. Current State of Research

Fehling et al. [4] present an optimization approach for the distribution of users to inter-dependent system components, such as servers or databases, in the context of multi-tenant SaaS applications. The focus lies on complete software systems, rather than discrete software services. Also, the authors do not regard different resource and VM types.

Andrzejak et al. [5] present, in the context of Amazon's *Spot Instance* VM auctioning system, a scheme for the determination of optimal bid prices, given a set of time-constrained but interruptible compute jobs. In this context, Andrzejak et al. consider various VM types, but no specific resource types. The focus of the work lies on the scheduling of jobs, rather than distribution.

Breitgand and Epstein [6] consider the optimal placement of VMs on physical machines in a data

center, with sets of VM executing a specific software service. Their work is focused on the role of a cloud provider, rather than a SaaS provider. Also, the authors neither consider predefined VM types nor specific resource types.

Kwok and Mohindra [7] present an approach for the optimal placement of multi-tenant SaaS applications in a data center under consideration of different resource types and QoS constraints. However, the authors do not specifically regard different VM types at varying price levels; accordingly, their objective consists of optimal resource exploitation on physical machines, rather than cost minimization.

In summary, none of the aforementioned approaches has explicitly defined and addressed the SSDP from the perspective of an SaaS provider in conjunction with specific resource types and different VM types yet. In addition, the majority of the previously outlined problem extensions have not been addressed by research to date, most notably not in conjunction.

IV. Research Objective: The Software Service Distribution Broker

My research aims at the design and implementation of an integrated *Software Service Distribution Broker*, which addresses the SSDP in conjunction with the various extensions that have been outlined. The broker will be accessible for SaaS providers and IaaS providers through a Web-based interface. Through the interface, the broker permits IaaS providers to register their VM type offers by submitting the relevant information regarding, e.g., resource supply, QoS guarantees, and price, in a structured form (*push* model). The tool will also allow to mine publicly available VM offer descriptions, e.g., through the Amazon EC2 API[1] (*pull* model). In addition, SaaS providers may specify the SSIs that have been requested by their end users. Based on this information, the Software Service Distribution Broker computes a *SSI distribution strategy* for the SaaS provider using suitable optimization algorithms. Thus, in accordance with the vision of a future cloud market [1], the broker facilitates the efficient distribution of SSIs through SaaS providers. An overview of the Software Service Distribution Broker is depicted in Figure 1.

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

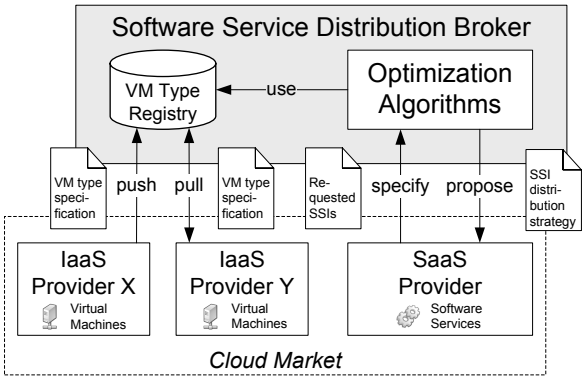

Figure 1. Overview of the Software Service Distribution Broker

V. Research Approach

In order to achieve my research objective, I will follow an approach that consists of the following four phases:

1) *Analysis*: Identification of the research problem and possible extensions. This includes a thorough analysis of current research and the identification of limitations or shortcomings in the state-of-the-art.
2) *Modeling & Development*: Conversion of the research problem and its extensions into formal, mathematical representations. Based on the resulting models, appropriate optimization approaches will be developed.
3) *Implementation*: Implementation of the optimization approaches and their integration into the proposed Software Service Distribution Broker.
4) *Evaluation*: Evaluation of the Software Service Distribution Broker and the incorporated optimization approaches, using mined data from real-life services and IaaS providers as a basis. The results serve as a basis for the validation of my research approach and objective.

Depending on the outcomes of the last phase, the process will, if necessary, be repeated multiply times.

VI. Preliminary Results

At the time of writing, I have completed a first iteration of the research process. This has resulted

in a formal model of the SSDP and a prototypical implementation and evaluation of an optimization approach, which will be presented in the following.

A. Formal Modeling

To address the SSDP through optimization algorithms, it has to be transformed into a formal mathematical model. To date, this modeling process has been completed for the basic form of the SSDP, as outlined in Section II-B.

The model is based on three basic assumptions: First, at least one suitable VM type (in terms of resource supply) exists for each requested SSI. In the worst case, one specific VM type is the sole suitable type for all SSIs. Thus, the *maximum* number of instances for each VM type corresponds to the number of SSIs. Second, an absence of SSI deployment costs is assumed. Third, SSIs may be arbitrarily combined on one VM instance, given that all resource demands are satisfied.

Let $S = \{1, ..., m\}$ be a set of SSI requests, $V = \{1, ..., n\}$ be a set of VM types, and $R = \{1, ..., o\}$ be a set of resources. For each requested SSI and resource, a resource demand is specified, i.e., $RD = \{RD_{11}, ..., RD_{mo}\}$. Furthermore, for each VM type, the cost $C = \{C_1, ..., C_n\}$, and for each VM and resource type, a specific resource supply $RS = \{RS_{11}, ..., RS_{no}\}$, is given. Employing these formalisms, we obtain Model 1.

The objective is to minimize the Total Cost (TC) in Equation 1. The TC is given by the number of utilized instances of each VM type (U_v), which is determined in Equations 2 and 3, multiplied by the respective VM cost. Equation 4 ensures that all SSIs are assigned to precisely one VM instance. Equation 5 guarantees that all resource constraints are held. Equation 6 defines x_{svi} as binary decision variables, which indicate whether a SSI s has been assigned to a VM of type v with the instance index i. Equation 7 further specifies the binary decision variables y_{vi}, which determine whether a certain instance i of VM type v is utilized. Because the maximum number of VM instances of each type corresponds to the number of SSIs, it generally holds that $i \in S$.

As can be observed, the formalization of the SSDP in Model 1 constitutes a *Linear Program* (LP). Thus, an optimal solution can be obtained using standard linear programming techniques, e.g., *brand and bound* [8].

Model 1 Software Service Distribution Problem

$$\text{Minimize } TC(x, y) = \sum_{v \in V} U_v * C_v \quad (1)$$

$$U_v = \sum_{i \in S} y_{vi} \quad \forall v \in V \quad (2)$$

$$y_{vi} \geq x_{svi} \quad \forall s \in S, v \in V, i \in S \quad (3)$$

$$\sum_{v \in V, i \in S} x_{svi} = 1 \quad \forall s \in S \quad (4)$$

$$\sum_{s \in S} RD_{sr} * x_{svi} \leq RS_{vr} \quad \forall v \in V, i \in S, r \in R \quad (5)$$

$$x_{svi} \in \{0, 1\} \quad \forall s \in S, v \in V, i \in S \quad (6)$$

$$y_{vi} \in \{0, 1\} \quad \forall v \in V, i \in S \quad (7)$$

B. Implementation & Preliminary Evaluation

Using the *lpsolve*[2] framework, I have implemented the LP-based optimization algorithm in a prototypical Java program, which constitutes a first step toward the proposed Software Service Distribution Broker.

In order to assess the proposed approach in terms of runtime performance, I have conducted a preliminary evaluation. For that matter, I created a set of SSDPs with a varying number of and requested SSIs (n_s), VM types (n_v), and resource types (n_r). For the VM and resource types, I used the specifications of the Amazon EC2 On-Demand VM offers in the European Union (resulting in 8 VM types and 3 resource types, namely processor, memory, and storage). In order to obtain realistic SSI execution data, I measured the absolute resource demands of 4 different contemporary video games on a local desktop computer.

Based on this information, I created 12 classes of SSDPs with fixed values for n_s, n_v, and n_r. For each class, 200 individual problems were generated, with the specific set of requested SSIs and regarded VM types and resource types being randomly selected. All resulting SSDPs were subsequently solved using the

[2]http://sourceforge.net/projects/lpsolve/

LP-based optimization algorithm, and the required computation time was measured.[3] Table I provides an overview of the evaluated SSDP classes and results: The column *solved* indicates the number of SSDPs that could be optimally solved by the program within a timeout period of 600 000 ms (10 minutes). The columns μ_t and $CI95_t$ provide the mean and (half width of the) 95% confidence interval of computation times for the resulting samples of *solved* SSDPs.

Table I
Computation Times for Optimal Distribution Strategies

n_s	n_v	n_r	solved	μ_t [ms]	$CI95_t$ [ms]
4	2	1	200	1.5	0.4
4	2	2	200	2.7	0.5
4	4	3	200	10.3	1.9
4	8	3	200	20.5	1.0
8	2	1	193	2 636.2	1 807.7
8	2	2	179	3 170.6	2 177.7
8	4	3	173	2 557.1	1 387.8
8	8	3	148	3 569.6	797.1
12	2	1	150	7 291.2	6 003.5
12	2	2	115	20 314.8	11 051.8
12	4	3	67	68 706.1	29 050.4
12	8	3	7	281 148.6	90 304.5

VII. Summary and Future Work

In the work at hand, I introduced the *Software Service Distribution Problem* as a novel research challenge in the context of cloud computing. The problem concerns the distribution of software service instances across available cloud infrastructure in the form of virtual machines. I have proposed the Software Service Distribution Broker, which facilitates the effective and (cost-)efficient service deployment through the computation of distribution strategies. I have formulated a basic optimization model and implemented a prototypical optimization algorithm. This algorithm has been evaluated, based on realistic virtual machine and software service execution data.

My future work will focus on two issues. The first is the detailed specification of the problem extensions that have been outlined in Section II-C, and their inclusion into the existing mathematical model and optimization approach. Second, the evaluation results indicate that the computation of optimal distribution strategies incurs substantial computational effort. In order to address real-life allocation problems that may contain a large number of software service instances and short planning cycles, suitable heuristics are required. Thus, my future work involves the development of such heuristics and their subsequent evaluation, concerning aspects such as runtime performance and solution quality.

[3]The evaluation was conducted on a dedicated laptop computer, equipped with an Intel Core i5-450M processor and 2 GB of memory.

Acknowledgments

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

References

[1] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[2] G. Briscoe and A. Marinos, "Digital Ecosystems in the Clouds: Towards Community Cloud Computing," in *3rd Int. Conf. on Digital Ecosystems and Technologies*, 2009, pp. 103–108.

[3] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A View of Cloud Computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[4] C. Fehling, F. Leymann, and R. Mietzner, "A Framework for Optimized Distribution of Tenants in Cloud Applications," in *3rd Int. Conf. on Cloud Computing*, 2010, pp. 252–259.

[5] A. Andrzejak, D. Kondo, and S. Yi, "Decision Model for Cloud Computing under SLA Constraints," in *18th Annual Meeting of the IEEE/ACM Int. Symp. on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*, 2010, pp. 257–266.

[6] D. Breitgand and D. Epstein, "SLA-aware Placement of Multi-Virtual Machine Elastic Services in Compute Clouds (TR H-0287)," IBM Research Division, Tech. Rep., 2010.

[7] T. Kwok and A. Mohindra, "Resource Calculations with Constraints and Placement of Tenants and Instances for Multi-Tenant SaaS Applications," in *6th Int. Conf. on Service-Oriented Computing*, 2008, pp. 633–648.

[8] F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*, 7th ed. McGraw-Hill, 2000.