# A General Approach to Service Deployment in Cloud Environments

Wubin Li, Petter Svärd, Johan Tordsson and Erik Elmroth
Department of Computing Science
Umeå University
SE-901 87 Umeå, Sweden
Email: {wubin.li, petters, tordsson, elmroth}@cs.umu.se

***Abstract*—The cloud computing landscape has recently developed into a spectrum of cloud architectures, leading to a broad range of management tools for similar operations but specialized for certain deployment scenarios. This both hinders the efficient reuse of algorithmic innovations within cloud management operations and increases the heterogeneity between different management systems. Our overarching goal is to overcome these problems by developing tools general enough to support the full range of popular architectures. In this contribution, we analyze commonalities in recently proposed cloud models (private clouds, multi-clouds, bursted clouds, federated clouds, etc.), and demonstrate how a key management functionality - service deployment - can be uniformly performed in all of these by a carefully designed system. The design of our service deployment solution is validated through demonstration of how it can be used to deploy services, perform bursting and brokering, as well as mediate a cloud federation in the context of the OPTIMIS Cloud toolkit.**

***Index Terms*—Cloud Computing; Cloud Architecture; Service Deployment;**

## I. Introduction

With the rise of virtualization as a platform for hosted services provision in the context of cloud computing, deployable cloud services are encapsulated in virtual appliances (VAs), and deployed by instantiating Virtual Machines (VMs) with virtual appliances [18]. This new manner of service deployment provides a direct route for traditional on-premises applications to be rapidly redeployed in a Software as a Service (SaaS [9]) mode. By decoupling the hardware and operating system of the infrastructure provider from the application stack provider, virtual appliances allow economies of scale on the one side to be leveraged by the economy of simplicity on the other.

In this paper, we present a novel approach to service deployment, general enough to meet the requirements of a range of common cloud scenarios, including private clouds, bursted clouds, federated clouds, multi-clouds, and brokered clouds. We identify the requirements for service deployment in these scenarios and present the architecture for a service deployment tool to meet these requirements. Our proposed tool interacts with components for, e.g., data management, service contextualization, service management in its orchestration of the service deployment process. Our proposed approach is validated by implementation and integration in a private cloud, a bursted cloud, and a brokered multi-cloud scenario using resources at Atos (Barcelona), BT (London), Flexiant (Edinburgh), and Umeå University (Umeå), as well as tools from the OPTIMIS Toolkit [16] providing the required complementing functionality.

The remainder of the paper is organized as follows: Section II introduces background cloud services and deployment. Section III describes core requirements for service deployment in cloud environments. Section IV presents the design of our service deployment solution. Section V describes a validation study of our approach in the context of OPTIMIS toolkit. Section VI outlines the related work of service deployment. Finally, our conclusions are presented in Section VII followed by a presentation of future work, acknowledgments, and a list of references.

## II. Background

### A. Cloud Services

Cloud services can be categorized into Software as a Service, Platform as a Service, and Infrastructure as a service, or SaaS, PaaS, and IaaS for short. In a cloud service deployment scenario the two stakeholders are the Infrastructure Provider (*IP*) and the Service Provider (*SP*). An IP offers infrastructure resources such as VMs, networks, and storage which can be used by SPs to deliver SaaS solutions to their customers. The SPs can also use PaaS tools to develop their services, or offer this functionality to their customers who may want to construct and deploy custom services. Without loss of generality, we concentrate in this contribution on the cases where an SP or an IP deploys services to an IP providing IaaS.

**Deployable Services**: Cloud systems offering IaaS are based on virtualization technology which means that a deployable cloud service is in fact a VM or a collection of VMs together with an description of the service, a *service manifest*. The manifest typically consists of sections describing what components the service is composed of along with functional and non-functional requirements for a deployment target. We refer to VM of a certain type as a *component* and note that a service can consist of multiple components. For example, a three-tier web application service may consist of a database component (e.g., MySql), an application component (e.g., Weblogic server [8]), and a presentation

layer component (e.g., Apache server). The service manifest can also define elasticity rules for the service, i.e. upper and lower bounds for how many instances of a component that are allowed to run. Commonly, associated with elasticity bounds are elasticity rules for when to scale up or down, which can range from simple *condition-action* statements to complex expressions that reason about statistical properties of the service workload. In addition, a service manifest typically contains various constraints such as desired geographical location, and data protection requirements, etc.

**The service lifecycle**: The lifecycle of a cloud service can be summarized as construction, deployment, operation, and undeployment. In the construction phase, the service applications (Virtual Appliances) are implemented and packaged into a set of VMs. The construction of the above discussed service manifest ends the service construction phase. The service deployment includes identification of a suitable deployment target, installation of the service VMs in the selected provider, and initialization of these VMs by the provider, i.e., VMs are booted, configured, and start to deliver the service. In the operation phase, the IP, and potentially also the SP, perform a set of management actions to ensure efficient and robust provisioning of the service. Once the service is no longer needed, it can be undeployed by the SP, upon which the IP shuts down the running VMs and removes any assets of the service. Notably, multiple instances of the same service can be created from a service manifest and these instances can be shutdown or restarted as needed.

### *B. Deployment scenarios*

Cloud environments can be set up differently depending on the types of interaction between the collaborating sites [27], [11]. For example, an SP can set up a cloud infrastructure for its own internal use, commonly referred to as a *private cloud*, which is illustrated in Figure 1. Private clouds can circumvent many of the security and privacy concerns related to hosted sensitive information in public clouds, the latter a case where the SP leases IaaS resources publicly available IPs. Private clouds may also offer stronger guarantees on control and performance as the whole infrastructure can be administered within the same domain.

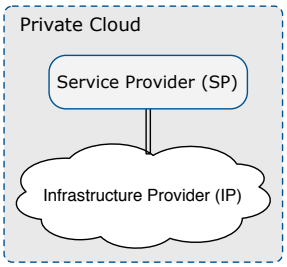


Fig. 1. Private cloud.

Private clouds may offload capacity to other IPs under periods of high workload, or for other reasons, e.g., planned maintenance of the internal servers. In this scenario, the providers form a hybrid architecture commonly referred to as a *cloud bursting* as seen in Figure 2. Typically, less sensitive tasks are executed in the public cloud instead while tasks that requiring higher levels of security are provisioned the private infrastructure.

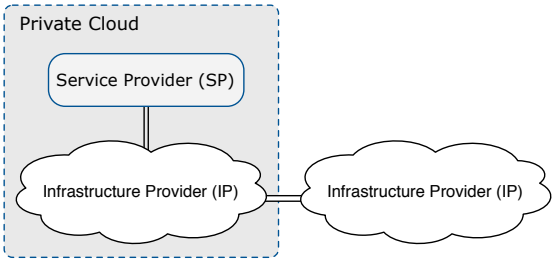


Fig. 2. Bursted (private) cloud.

*Federated clouds* are IPs collaborating on a basis of joint load-sharing agreements enabling them to offload capacity to each others [27] in a manner similar to how electricity providers exchange capacity. The federation takes place at the IP level in a transparent manner. In other words, an SP that deploys services to one of the IPs in a federation is not notified if its service is off-loaded to another IP within the federation. However, the SP is able to steer in which IPs the service may be provisioned, e.g., by specifying location constraints in the service manifest, Figure 3 illustrates a federation between three IPs.

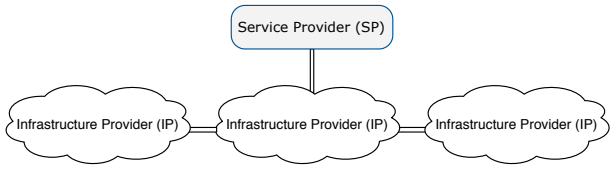


Fig. 3. Cloud federation.

If the IP itself is involved in selecting which IP a service should be deployed or re-deployed to the scenario is known as a *multi-cloud*. In multi-cloud deployments, such as in Figure 4, the SP is responsible for planning, initiating and monitoring the execution of services. Notably, we are implicitly considering split deployment scenarios, i.e., when the components of the service are deployment across multiple IPs.

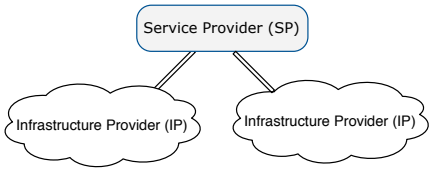


Fig. 4. Multi-cloud scenario.

A related scenario is when a *cloud broker* [32] handles the complexity of prioritization and selection of IPs, and may also offer value-add services to IPs and SP. In this case, the broker

has agreements with a number of IPs and selects the best match for a service based on the SP's desired criteria. The broker operates between the SP and the IPs, offering an IP-like interface to SPs and an SP-style one to IPs, as illustrated in Figure 5.

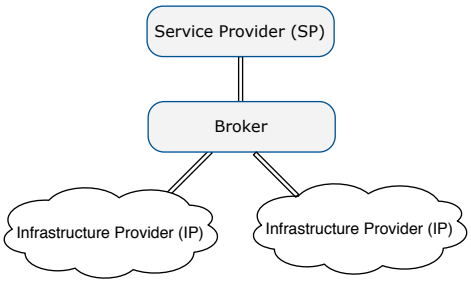


Fig. 5. Brokered scenario.

## III. Requirements for Service Deployment

Based on the service lifecycle and the various cloud architectures discussed in Section II, we identify the below list of requirements for service deployment. Notably, the order and exact details of what is performed in each step of the service deployment process may vary with the deployment scenario, but the following tasks are always performed.

- *Discovery of IPs*. This step is about identifying IPs that are available for deployment. IPs can be discovered by looking them up in a registry or by using auto-discovery mechanisms. We remark that discovery (along with the later filtering and selection) of an IP is trivial in the private cloud case, as a single IP is available.
- *Filtering of available IPs*. In order not to add overhead by negotiating deployment with IPs that fail to fulfill fundamental requirements for the particle service to be deployed, an initial filtering of the list of IPs retrieved during IP discovery must be possible. Criteria for filtering include both functional aspects, e.g., support for certain hypervisors and VM image formats, as well as non-functional criteria such as constraints based on the country in which the IP is based (for legal and/or data-protection reasons).
- *Construction of deployment descriptor*. Each service must be defined a service description that specifies the functional and non-functional parameters of the service. A service description is an abstract definition of the service, which is used to negotiate with IPs and later becomes part of the service agreement with the IP. Data specified in the service description, i.e. VM disk images, must also be prepared. A set of utilities for creation, modification, etc. of service manifests would greatly simplify this procedure.
- *Negotiation and deployment optimization*. It must be possible for an SP to negotiate with available IPs and ask these to provide offers for hosting the service (or parts of it). Based on the results these negotiations and data such as reputation statistics that could be evaluated by a third-party entities, the SP must be able to make a decision about where to deploy the service.
- *Service contextualization*. Before a service can be deployed, some information required to launch the service successfully, which is not known at the moment of VM image generation must be propagated to the IP. A possible mechanism for this *contextualization* process is to embed various scripts in the VM images that upon boot dynamically retrieves information such as network parameters, security credentials, etc., enabling the VM to self-contextualize.
- *Service data transfer.* It must be possible to transfer the contextualized VM images along with any other data required by the service to the IP. To be able to guarantee properties such as confidentiality and integrity of data during this transfer, a set of security mechanisms are required.
- *SLA creation*. To ensure that the service operates according to the SP's expectations, it must be possible to establish and SLA that governs the relationship between the SP and IP for the provisioning of the service. Penalties may be agreed upon in the case of non-compliance with the SLA. An SLA for service provisioning commonly includes segments that address: a definition of the service, performance measurements, problem management, customer duties, warranties, disaster recovery, as well as conditions for termination of the agreement [1].

Notably, these requirements for service deployment have significant similarities with the tasks identified in the overall process for resource selection (scheduling) in Grid computing environments [29].

## IV. SDO architecture

Based on the requirements study in the previous sections, we derive a general sequence for service deployment, containing the tasks to be performed. In this process, illustrated by the sequence diagram in Figure 6, the complexity of each task vary with the deployment scenario (private cloud, federation, bursting, etc.).

Notably, Step 1, service construction, is identifical no matter the scenario, an SP constructs (implements, packages, etc.) a service the same way no matter how it will be deployed. Step 2, identification and filtering of suitable IPs, is trivial for SPs in the private cloud case, as there is a single, well-known IP available. This step is also relatively easy in cloud brokering scenarios, but more difficult in federation and multi-cloud cases.

Most of the algorithmic complexity in service deployment is associated within the related tasks of SLA negotiation and IP assessment (Steps 3 and 4 in Figure 6). For the scenarios where the SP interacts with a single provider (the private cloud IP or the broker), these tasks are simplified. Conversely, for federation, bursting, multi-cloud deployments, interaction with more

than one IP complicates the process. The richness of the negotiation protocol can range from simple versions with primitives such as offer, accept, and refuse, to more complicated versions with counter-offers, etc., to approaches based on auctions. An in-depth analysis of negotiation protocols is beyond the scope of this paper. Further details on this topic are given, e.g., by Sarangan et al. [28] and Jennings et al. [17]. Similarly, for IP assessment, the complexity of estimating the utility associated with deploying the service in each potential provider can differ significantly based on the modeling method used. Algorithms proposed for optimizing provider selection include scheduling-inspired combinatorial optimization approaches such as integer programming, which are commonly suggested [14], [32], but tend to scale very poorly with the number of IPs. Others recommend heuristic solutions [23] that trade optimality for faster decision-making.

Once the most suitable provider (or potentially, set of providers in the multi-cloud case) is identified, the SP performs contextualization (Step 5 in the sequence diagram) to prepare the service VM images with any dynamic information that is needed for these to boot and configure themselves properly. This step may be more complicated if split deployment is performed for multi-clouds, as an external rendezvous mechanism typically is required in order to initialize cross-provider networking for the VMs of the service.

After VM images are properly configured, these are uploaded to the target provider(s) as illustrated in Step 6 of Figure 6. As VM images typically are very large files, significant performance gains can be achieved by proper tuning of network parameters. In private clouds where a network file system may connect IP and SP, image transfer is much less of an issue. Alternatively, if some public IP does not support upload of SP-defined VM images, a custom service image must be pre-created (based on templates from the provider) and stored at that IP. In such a case, contextualization abilities are significantly reduced.

When the contextualized VM images are stored in the IP, the SP confirms the offered negotiated in Step 3 and an SLA is created between the SP and the IP for the provisioning of the, as illustrated in Step 7. Once again, this step is more complex for multi-clouds, where the SP need to aggregate multiple SLAs from different IPs.

Finally, Step 8 in Figure 6 illustrates that once the service is deployed, the SP stores some state information about it, to enable subsequent service monitoring, management, and undeployment.

To fulfill the requirements of service deployment and perform the steps in Figure 6, we propose a Service Deployment Optimization (SDO) architecture. The purpose of the SDO tool is two-fold - it is responsible for generating optimal deployment solutions for a service, and for coordinating the deployment process in order to provision a service in IP(s) according to the deployment plan. In order to separate the placement optimization from the deployment coordination functionality, the SDO is split into two components, the *Service Deployer* (SD), and the *Deployment Optimizer* (DO),

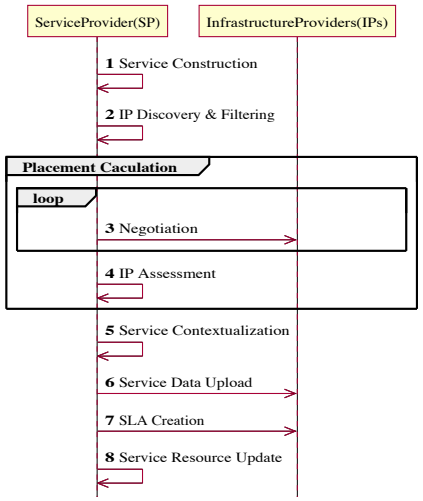


Fig. 6. Sequence Diagram.

both illustrated in Figure 7. The DO is a decision-making component and the SD is a module that orchestrates the DO and various utility functionalities in order to perform the deployment sequence described in Figure 6.

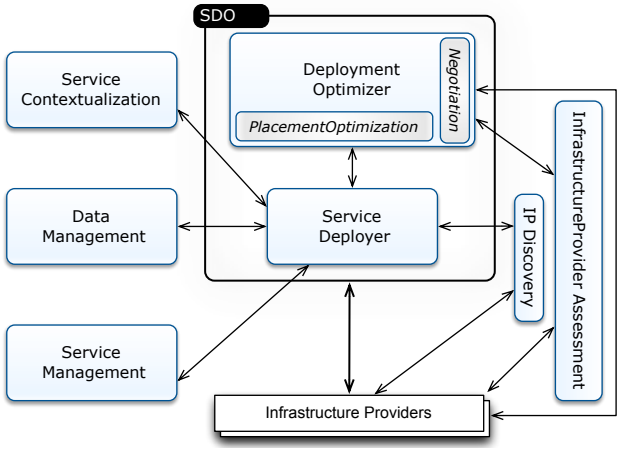


Fig. 7. Overview of SDO architecture.

We outline the main design rationale for the SD and DO components below, as well as discuss how they interact with each other and related utility functionalities for data transfer, etc.

### A. Service Deployer

The SD is designed to coordinate the deployment and interact with the other involved parties in a deployment. The SD takes a service deployment request, contacts the IP discovery service to obtain which providers are available and performs filtering (see steps 1-2 in Figure 6). To retrieve an optimal placement scheme, SD contacts the DO who performs calculation for placement optimization. Once an optimal placement solution is returned, the SD deploys a whole service following steps 5-8 in Figure 6 with the support of

external components. *Service Contextualization* is in charge of contextualizing VM images, *Data Management* is responsible for data transfer from the SP side to the IP side, *Service Management* creates service resource and updates resource accordingly, and *SLA Management* handles the IP side creation of agreement.

### B. Deployment Optimizer

The DO's inputs from the SD include a service manifest, the optimization objective, and available IP info, etc. Based on those parameters, the DO generates an optimal placement scheme for the service. In order to achieve an optimal placement objective, the DO may split services that contains more than one component into several sub-services, and map them to different IPs. This is provided it can do so without breaking affinity constraints specified in the service description, During the calculation, the DO negotiates with IPs and the IP assessment tools, see steps 3-4 in Figure 6. Optimization techniques such as combinatorial optimization, problem relaxations and heuristic approaches such as greedy formulation can be applied in this component.

## V. Validation Study

In order to verify that our service deployment architecture is suitable for the envisioned cloud architectures, we perform a validation study. The study is carried out in the context of the OPTIMIS Toolkit [16], which includes a set of independent components that can be adopted, either in full or in part, by IPs that provide infrastructure resources, and by SPs that use these capacities to deliver services. The study comprises three cloud service deployment scenarios: private cloud, cloud brokerage, and cloud bursting.

In these three scenarios, the service we use for validation is a composite service for gene detection presented in [31]. This service contains five components. First, there are four functionality components which contribute to the overall gene detection process: translation of the input genomic database to a given format (component GA); obtention of a list of aminoacid sequences which are similar to a reference input sequence (component GB); search of the relevant regions of the genomic database (component GC) and execution of the GeneWise [13] algorithm on them (component GD). Additionally, there is one component for coordination (component GP). Each component can be encapsulated in a VM sized approx 9.8 GB. To avoid repetitive data transmission, only one VM image is transferred from SP to IP if there are multiple components deployed to the IP, while in this case multiple VM instances are to be started using the same image with different contextualized data.

For the validation, we use a distributed testbed with four IPs located across Europe: Atos [2] (Spain), BT [3] (UK), Flexiant [4] (UK) and Umeå University (Sweden). Each IP site hosts selected parts of the OPTIMIS Toolkit, as well as fundamental management software such as Xen [5] and Nagios [6]. The role of the IPs in the different scenarios is summarized in Table I. Notably, our goal is not to evaluate the various providers but rather to investigate how well our proposed approach adapt to real scenarios.

TABLE I
USE CASE CONFIGURATIONS

| | Atos | BT | Umeå University | Flexiant |
|---|---|---|---|---|
| Private Cloud | ✓ | | | |
| Cloud Brokerage | ✓ | ✓ | ✓ | ✓ |
| Cloud Bursting | ✓ | | | ✓ |

### A. SDO in OPTIMIS

In the integration with OPTIMIS, the SDO interacts with the following external components altogether providing the functionality of the external components illustrated in Figure 7.

- *IP Discovery:* IP information is registered in a simple online registry accessed through a REST interface. In this registry, information such as IP identifier, IP name, and endpoints for negotiation, etc., are stored.
- *VM Contextualization*: This component provides an interface for constructing service context data, such as security certificates, VPN hostnames, VPN DNS and Gateway IP addresses, mount points for network data stores, monitoring manager hostnames, off-line software license tokens and a list of software dependencies [12].
- *Data Manager*: A Front-end, Hadoop-based [7] Data Management service enriched with RESTful APIs in front of Hadoop and a series of components that aim to extend Hadoop's functionality beyond its well known back-end, heavy data processing scope [21].
- *SLA Management*: A service and client based on WS-Agreement protocol [10] for negotiating and creating Service Level Agreements between IP and SP [22].
- *Infrastructure Provider Assessment*: In OPTIMIS, deployment decision is based on four key factors - trust, risk, eco-efficiency and cost (TREC). TREC parameters are used by DO to perform IP assessments.

### B. Scenarios Descriptions and Statistics

- Private Cloud

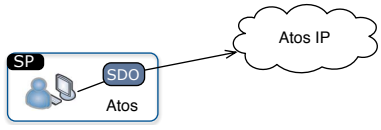


Fig. 8. Private scenario.

In the Private Cloud scenario, the SP (also located in the Atos cloud) submits the gene detection service deployment request to the Atos cloud. All components (GA, GB, GC, GD, and GP) are deployed to the Atos IP.

- Cloud Brokerage (Multi-Cloud)
In the Cloud Brokerage scenario, two SDO instances are running: one in the Umeå cloud, which plays the role of an SP. The other one is located in the BT cloud, which plays the role of a Cloud broker. The SP submits the

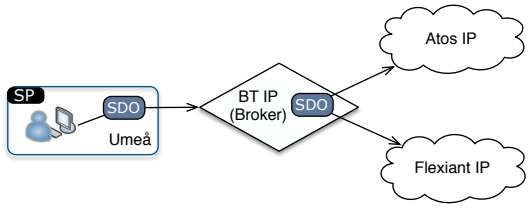


Fig. 9. Cloud brokerage scenario.

gene detection service deployment request to the Umeå cloud. Instead of deploying the service by itself, the SDO in the Umeå cloud calls the SDO on the broker to complete the deployment. There are three IPs registered in the IP registry which can be queried by the SDO in the BT cloud. After the by Deployment Optimizer's calculations (including IP assessment, negotiation, and placement optimization) two IPs are selected to host the service. Specifically, two components (GC and GD) are to be deployed to Flexiant cloud, the other three (GA, GB, and GP) are to be deployed to the Atos cloud. For the purpose of this demonstration, VM images are stored on the broker in advance.

- Cloud Bursting

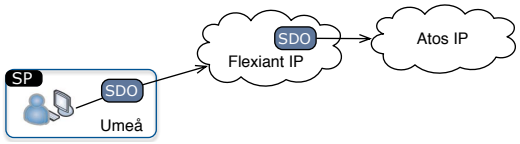


Fig. 10. Cloud bursting scenario.

In the Cloud Bursting scenario, the service is already deployed in the Flexiant cloud. To fulfill a demand for increasing service capacity from the SP, the Flexiant cloud needs to launch two more instances respectively for two of the five components (i.e., GC and GD) in the service. For financial reasons, the Flexiant cloud decides to outsource this demand to a more cheaper cloud provider, i.e., Atos cloud, while maintaining its SLA-agreement with SP.

### C. Experimental results

In order to assess the performance of the SDO and the complexity of the service deployment process as such, we measure the duration of the main steps of deployment for each studied cloud architecture. Table II presents statistics of time consumed in each phase of service deployment for each scenario.

From this table, we conclude that the major part of the time is used to transfer VM images from the SP to the IP. Notable, the differences in image transfer time among the scenarios are due to the complexity of the placement scheme. For the private cloud scenario, all components are deployed to Atos. Only one VM image needs to be transferred internally. For Multi-Cloud scenario, two VM images are transferred, one from BT to Atos (for components GA, GB, and GP), the other one from BT to Flexiant (for components GC and GD).

TABLE II
ILLUSTRATIVE PERFORMANCE RESULTS FOR THE DEMONSTRATED DEPLOYMENT SCENARIOS (SECONDS)

| Deployment Phases | Private | Multi-Cloud | Cloud Bursting |
|---|---|---|---|
| IP Discovery | 0 | 2 | 1 |
| Placement Calculation | 2 | 108 | 13 |
| VM Contextualization | 11 | 19 | 15 |
| Data Upload | 598 | 1546 | 701 |
| Service Resource Creation | 4 | 4 | 2 |
| Agreement Creation | 12 | 23 | 17 |

Another observation is that placement calculation becomes more complex in the multi-cloud case, where the number of potential service configurations is much larger than for the private and bursting cases. During the brokering case, multiple negotiations are performed between BT cloud and Atos cloud, and Flexiant cloud for cost inquiry. In addition, IP assessment is also based on IP evaluations in terms of trust, risk-level, and eco-efficiency which are independently verified by querying a trusted database containing historical information pertaining to these factors .

In summary, the Private Cloud scenario demonstrates how the SDO can be used to complete a service deployment in general. The Cloud Brokerage scenario demonstrates cloud brokerage and federation across multiple cloud providers. The Cloud Bursting scenario shows how organizations can utilize the SDO to scale out their infrastructure, using resources from third-party providers based upon a range of factors such as trust, risk assessment [20], eco-efficiency and cost.

## VI. Related Work

Talwar et al. [30] review approaches for service deployment before the emergence of Cloud Computing. They compare and evaluate four types of service-deployment approaches, i.e., manual, script-, language-, and model-based solutions, in terms of scale, complexity, expressiveness, and barriers for first usage. They also conclude that service deployment technologies based on scripts and configuration files have limitation to express dependencies and verify configurations, and usually result in erroneous system configurations, while language- and model-based approaches address these challenges with comparatively higher barriers for first usage.

With the emergence of Cloud Computing, services are provisioned using virtual machines. Service deployment can be done by initializing virtual machines with their virtual appliances. Cloud users are enabled to deploy applications without confronting the usual obstacles of maintaining hardware and system configurations. Lots of work have been done in the context of this new service-deployment technology. Most of these are focusing on approaches to optimization, e.g., Kecskemeti et al. [19] who propose an automated virtual appliance creation service that aids the service developers to create efficiently deployable virtual appliances. They reduce deployment time of the service by rebuilding the virtual appliance of the service on the deployment target site. For optimal

virtual machine placement across multiple cloud providers, Chaisiri et al. [14] propose an stochastic integer programming (SIP) based algorithm that can minimize the cost spent in each placement plan for hosting virtual machines in a multiple cloud provider environment under future demand and price uncertainty. Similarly, Vozmediano et al. [26] [25] explore the multi-cloud scenario to deploy a computing cluster on top of a multi-cloud infrastructure, for solving loosely-coupled Many-Task Computing (MTC) applications. In this way, the cluster nodes can be provisioned with resources from different clouds to improve the cost-effectiveness of the deployment, or to implement high-availability strategies.

Our previous contributions in this field include a cloud brokering mechanisms [32] for optimized placement of VMs to obtain optimal cost-performance tradeoffs across multiple cloud providers in static scenarios, and a linear programming model to dynamically reschedule VMs (including modeling of VM migration overhead) upon changed conditions such as price changes, service demand variation, etc. in dynamic cloud scheduling scenarios [24], as well as an approach to optimal virtual machine placement within datacenters for predicable and time-constrained load peaks [23].

However, although algorithms for optimizing service deployment is a very active area of research, and a lot of interest is given to the various deployment architectures in general, we have not been able to identify any results on the topics of this contribution, namely architectures and tools general enough to support all current deployment scenarios.

## VII. Conclusion and Future Work

In this paper, we propose a general approach to automatic service deployment in cloud environments, based on our study of cloud architectures and deployment scenarios and the core requirements for service deployment derived from these. A validation study performed in the context of the OPTIMIS Toolkit verifies the feasibility of a general service deployment component that can be reused across multiple cloud architectures. Our validation study also gives some indications about the performance aspects of cloud service deployment, identifying transfer of VM images as the most time-consuming task.

Future directions for this work includes in-depth studies of algorithms for optimized selection of deployment targets. Another topic of future research is the incorporation of *re-deployment*, i.e., migration of the full service, or some of its components, to other IP(s) during operation [15]. Reasons for re-deployment include improved performance, and improved cost-efficiency. In such scenarios, a careful tradeoff between re-deployment overhead and expected improvement must be considered [24]. Additionally, a model of interconnection requirements that can precisely express the relationships between components within a service to be deployed can be another promising direction to investigate. Such a model can help SDO optimizing the service deployment with e.g., less communication cost between service components. In addition, we are working on a specific scenario where cloud users can specify hard constraints and soft constraints when demanding resource provisions. A hard constraint is a condition that has to be satisfied when deploying services, i.e., it is mandatory. In contrast, a soft constraint (also called a preference) is optional. An optimal placement solution with soft constraints satisfied is preferable over other solutions. We are also investigating how to apply multi-objective optimization techniques to this scenario.

## Acknowledgments

We acknowledge the members of the OPTIMIS team for their valuable help in integration and testing for the validation scenario. We also thank Atos, BT, and Flexiant for providing servers for the distributed testbed. Financial support has in part been provided by the European Community's Seventh Framework Programme ([FP7/2010-2013]) under grant agreements no. 257115 (OPTIMIS [16]), and the Swedish Government's strategic effort eSSENCE.

## References

[1] An outline of the core elements of an SLA, http://www.sla-zone.co.uk/, visited May 2012.
[2] Atos, 2012, http://atos.net, visited May 2012.
[3] BT Group, 2012, http://www.bt.com/, visited May 2012.
[4] Flexiant Cloud, 2012, http://www.flexiant.com/, visited May 2012.
[5] Xen, 2012, http://www.xen.org/, visited May 2012.
[6] Nagios, 2012, http://www.nagios.org/, visited May 2012.
[7] Apache Hadoop Project. http://hadoop.apache.org/, visited May, 2012.
[8] Oracle WebLogic Server. http://www.oracle.com/technetwork/middleware/weblogic/overview/index.html, visited May, 2012.
[9] Software as a Service (SaaS). Cloud Taxonomy. http://cloudtaxonomy.opencrowd.com/taxonomy/software-as-a-service/, visited May, 2012.
[10] Web Services Agreement Specification (WS-Agreement). http://www.ogf.org/documents/GFD.107.pdf, visited May, 2012.
[11] M. Ahronovitz et al. Cloud computing use cases white paper, v4.0. www.cloudusecases.org, visited May 2012.
[12] D. Armstrong, K. Djemame, S. Nair, J. Tordsson, and W. Ziegler. Towards a Contextualization Solution for Cloud Platform Services. In *Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science*, CloudCom'11, pages 328–331, Washington, DC, USA, 2011. IEEE Computer Society.
[13] E. Birney, M. Clamp, and R. Durbin. GeneWise and Genomewise. *Genome Research*, 14(5):988–995, May 2004.
[14] S. Chaisiri, B.-S. Lee, and D. Niyato. Optimal virtual machine placement across multiple cloudproviders. In *Proceedings of the 4th IEEE Asia-Pacific Services Computing Conference*, pages 103–110.
[15] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *NSDI '05*, pages 273–286. ACM, May 2005.
[16] A. J. Ferrer, F. Hernández, J. Tordsson, E. Elmroth, A. Ali-Eldin, C. Zsigri, R. Sirvent, J. Guitart, R. M. Badia, K. Djemame, W. Ziegler, T. Dimitrakos, S. K. Nair, G. Kousiouris, K. Konstanteli, T. Varvarigou, B. Hudzia, A. Kipp, S. Wesner, M. Corrales, N. Forgó, T. Sharif, and C. Sheridan. OPTIMIS: A Holistic Approach to Cloud Service Provisioning. *Future Generation Computer Systems*, 28(1):66–77, 2012.
[17] N. Jennings, P. Faratin, A. Lomuscio, S. Parsons, M. Wooldridge, and C. Sierra. Automated negotiation: prospects, methods and challenges. *Group Decision and Negotiation*, 10(2):199–215, 2001.
[18] G. Kecskemeti, P. Kacsuk, T. Delaitre, and G. Terstyanszky. Virtual Appliances: A Way to Provide Automatic Service Deployment. In F. Davoli, N. Meyer, R. Pugliese, and S. Zappatore, editors, *Remote Instrumentation and Virtual Laboratories*, pages 67–77. Springer US, 2010.
[19] G. Kecskemeti, G. Terstyanszky, P. Kacsuk, and Z. Neméth. An Approach for Virtual Appliance Distribution for Service Deployment. *Future Gener. Comput. Syst.*, 27(3):280–289, March 2011.

[20] M. Kiran, M. Jiang, D. J. Armstrong, and K. Djemame. Towards a Service Lifecycle Based Methodology for Risk Assessment in Cloud Computing. In *Proceedings of the 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, DASC'11, pages 449–456, Washington, DC, USA, 2011. IEEE Computer Society.

[21] G. Kousiouris, G. Vafiadis, and T. Varvarigou. A Front-end, Hadoop-based Data Management Service for Efficient Federated Clouds. In *Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science*, CloudCom'11, pages 511–516, Washington, DC, USA, 2011. IEEE Computer Society.

[22] A. Lawrence, K. Djemame, O. Wäldrich, W. Ziegler, and C. Zsigri. Using Service Level Agreements for Optimising Cloud Infrastructure Services. In *Proceedings of the 2010 international conference ServiceWave*, ServiceWave'10, pages 38–49, Berlin, Heidelberg. Springer-Verlag.

[23] W. Li, J. Tordsson, and E. Elmroth. Virtual Machine Placement for Predictable and Time-Constrained Peak Loads. In *Proceedings of the 8th international conference on Economics of grids, clouds, systems, and services (GECON'11)*. Lecture Notes in Computer Science, Vol. 7150, Springer-Verlag, pp. 120-134, 2011.

[24] W. Li, J. Tordsson, and E. Elmroth. Modeling for Dynamic Cloud Scheduling via Migration of Virtual Machines. In *Proceedings of the 3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2011)*, pages 163–171, 2011.

[25] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente. Elastic management of web server clusters on distributed virtual infrastructures. *Concurrency and Computation: Practice and Experience*, 23(13):1474–1490, 2011.

[26] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente. Multicloud deployment of computing clusters for loosely coupled mtc applications. *IEEE Transactions on Parallel and Distributed Systems*, 22:924–930, 2011.

[27] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, and F. Galan. The RESERVOIR model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53(4):1–11, 2009.

[28] V. Sarangan and J. Chen. Comparative study of protocols for dynamic service negotiation in the next-generation internet. *Communications Magazine, IEEE*, 44(3):151–156, 2006.

[29] J. Schopf. Ten actions when grid scheduling. In *Grid resource management*, pages 15–24. Kluwer Academic Publishers, 2003.

[30] V. Talwar, D. Milojicic, Q. Wu, C. Pu, W. Yan, and G. Jung. Approaches for Service Deployment. *IEEE Internet Computing*, 9(2):70–80, Mar. 2005.

[31] E. Tejedor, J. Ejarque, F. Lordan, R. Rafanell, J. Alvarez, D. Lezzi, R. Sirvent, and R. Badia. A cloud-unaware programming model for easy development of composite services. In *2011 Third IEEE International Conference on Coud Computing Technology and Science*, pages 375–382. IEEE, 2011.

[32] J. Tordsson, R. Montero, R. Moreno-Vozmediano, and I. Llorente. Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. *Future Generation Computer Systems*, 28(2):358–367, 2012.