Introduction to Track on Engineering Virtualized Services*

Reiner Hähnle[1] and Einar Broch Johnsen[2]

[1] Technical University of Darmstadt, Germany
haehnle@cs.tu-darmstadt.de
[2] Dept. of Informatics, University of Oslo, Norway
einarj@ifi.uio.no

Abstract. Virtualization is a key technology enabler for cloud computing. Despite the added value and compelling business drivers of cloud computing, this new paradigm poses considerable new challenges that have to be addressed to render its usage effective for industry. Virtualization makes elastic amounts of resources available to application-level services; for example, the processing capacity allocated to a service may be changed according to demand. Current software development methods, however, do not support the modeling and validation of services running on virtualized resources in a satisfactory way. This seriously limits the potential for fine-tuning services to the available virtualized resources as well as for designing services for scalability and dynamic resource management. The track on *Engineering Virtualized Services* aims to discuss key challenges that need to be addressed to enable software development methods to target resource-aware virtualized services.

1 Moving into the Clouds

The planet's data storage and processing is about to move into the clouds. This has the potential to revolutionize how we will interact with computers in the future. Although the privacy of data stored in the cloud remains a challenge, cloud-based data processing, or cloud computing, is already emerging as an economically interesting business model, due to an undeniable added value and compelling business drivers [5]. One such driver is *elasticity*: businesses pay for computing resources when they are needed, instead of provisioning in advance with huge upfront investments. New resources such as processing power or memory can be added to the cloud's virtual computers on the fly, or additional virtual computers can be provided to the client application. Going beyond shared storage, the main potential in cloud computing lies in its scalable virtualized framework for data processing. If a service uses cloud-based processing, its capacity can be automatically adjusted when new users arrive. Another driver is *agility*: new services can be deployed quickly and flexibly on the market at limited

* Partly funded by the EU project FP7-610582 ENVISAGE: Engineering Virtualized Services (http://www.envisage-project.eu).

cost. This allows a service to handle its end-users in a flexible manner without requiring initial investments in hardware before the service can be launched.

Reliability and control of resources are barriers to the industrial adoption of cloud computing today. To overcome these barriers and to gain control of the virtualized resources on the cloud, client services need to become resource-aware. Looking beyond today's cloud, we may then expect *virtualized services* which dynamically combine distributed and heterogeneous resources from providers of utility computing in an increasingly fine-grained way. Making full usage of the potential of virtualized computation requires that we rethink the way in which we design and develop software.

2 Empowering the Designer

The elasticity of software executed in the cloud means that its designers are given far reaching control over the resource parameters of the execution environment, such as the number and kind of processors, the amount of memory and storage capacity, and the bandwidth. In principle, these parameters can even be changed dynamically, at runtime. This means that the client of a cloud service not only can deploy and run software, but is also in full control of the trade-offs between the incurred cost and the delivered quality-of-service.

To exploit these new possibilities, software in the cloud must be *designed for scalability.* Today, software is often designed based on specific assumptions about deployment, such as the size of data structures, the amount of random access memory, the number of processors. Rescaling usually requires extensive design changes when scalability has not been taken into account from the start. This consideration makes it clear that it is essential to detect and fix deployment errors, such as the impossibility to meet a service level agreement, already *in the design phase.* To make full usage of the opportunities of cloud computing, software development for the cloud demands a design methodology that

– can take into account deployment modeling at early design stages and
– permits the detection of deployment errors early and efficiently, preferably using software tools, such as simulators, test generators, and static analyzers.

Clearly, there is a new *software engineering challenge* which needs to be addressed: how can the validation of deployment decisions be pushed up to the modeling phase of the software development chain without convoluting the design with deployment details?

3 Controlling Deployment in the Design Phase

When a service is developed today, the developers first design its *functionality*, then they determine which resources are needed for the service, and ultimately the *provisioning* of these resources is controlled through a *service level agreement* (SLA). So far, these three parts of a deployed cloud service tend to live in separate worlds. It is important to bridge the gaps between them.

The first gap is between the client layer functionality and the provisioning layer. It can be closed by a virtualization interface which allows the client layer to read and change resource parameters. The second gap is between SLAs and the client layer. Here the key observation is that the service contract part of an SLA can be formalized as a specification contract with rigorous semantics. This enables formal analysis of the client behavior with respect to the SLA *at design time*. Possible analyses include resource consumption, performance analysis, test case generation, and formal verification [2]. For suitable modeling and specification languages such analyses can be highly automated [3].

4 The Papers

The ISoLA track *Engineering Virtualized Services*, organized in the context of the EU FP7 project Envisage, reflects the aims laid down above and focuses on systematic and formal approaches to

– modeling services deployed on the cloud,
– formalizing SLAs, and
– analysis of SLAs.

A crucial aspect in modeling cloud services is the handling of faults and errors. This is addressed in the papers by Göri et al. [9] and Lanese et al. [10] The former critically discusses the different choices that have to be made when defining a fault model for concurrent, actor-based languages. The latter proposes a specific failure model for concurrent objects with cooperative scheduling that automatically re-establishes object invariants after program failures, thereby eliminating the need to manually write this problematic code. The paper by De Boer & Nobakht [7] shows that high-level modeling of services can be achieved already on the Java level by embedding an actor-based API with the help of lambda expressions and extended dynamic invocation support, which is available since Java 8.

There are two papers on formalization of SLAs and service contracts: the paper by Woodcock et al. [11] describes the COMPASS Modelling Language CML, which is used to formally model large-scale Systems of Systems and the contracts which bind them together. The paper by Causevic et al. [6] presents the REMES HDCL language by way of a case study that formalizes service negotiation in a distributed energy management scenario.

Moving to analysis, a central aspect in deployment of cloud services is to obtain reliable estimates on resource consumption and, hence, on adequate provisioning. The paper by Giachino & Laneve [8] suggests a type system for a concurrent, object-oriented language that permits dynamic scaling out and scaling in. The type of a program is behavioural and it reflects the resource deployments over periods of time. On the other hand, the paper by Albert et al. [1] focuses on automatic inference of upper bounds for the amount of data transmissions that may occur in a distributed system. Finally, the paper by Bubel et al. [4] concentrates on formal verification of functional aspects of service contracts: it presents a technique for compositional verification in presence of constant evolutionary changes to the verification target.

Taken together, the eight papers in this track comprise an exciting snapshot of the state-of-art in formal approaches to modeling services deployed on the cloud as well as to formalization and analysis of SLAs.

References

1. E. Albert, J. Correas, E. Martin-Martin, and G. Roman-Diez. Static inference of transmission data sizes in distributed systems. In T. Margaria and B. Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation, 6th International Symposium, ISoLA 2014, Corfu, Greece*, LNCS. Springer, Oct. 2014. In this proceedings.
2. E. Albert, F. de Boer, R. Hähnle, E. B. Johnsen, and C. Laneve. Engineering virtualized services. In M. A. Babar and M. Dumas, editors, *2nd Nordic Symposium on Cloud Computing & Internet Technologies (NordiCloud'13)*, pages 59–63. ACM, 2013.
3. E. Albert, F. de Boer, R. Hähnle, E. B. Johnsen, R. Schlatte, S. L. Tapia Tarifa, and P. Y. H. Wong. Formal modeling of resource management for cloud architectures: An industrial case study. *Journal of Service-Oriented Computing and Applications*, 2013. Springer Online First, DOI 10.1007/s11761-013-0148-0.
4. R. Bubel, R. Hähnle, and M. Pelevina. Fully abstract method contracts. In T. Margaria and B. Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation, 6th International Symposium, ISoLA 2014, Corfu, Greece*, LNCS. Springer, Oct. 2014. In this proceedings.
5. R. Buyya, C. S. 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*, 25(6):599–616, 2009.
6. A. Causevic, C. Seceleanu, and P. Pettersson. Distributed energy management case study: A formal approach to analyzing utility functions. In T. Margaria and B. Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation, 6th International Symposium, ISoLA 2014, Corfu, Greece*, LNCS. Springer, Oct. 2014. In this proceedings.
7. F. de Boer and B. Nobakht. Programming with actors in Java 8. In T. Margaria and B. Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation, 6th International Symposium, ISoLA 2014, Corfu, Greece*, LNCS. Springer, Oct. 2014. In this proceedings.
8. E. Giachino and C. Laneve. Towards the typing of resource deployment. In T. Margaria and B. Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation, 6th International Symposium, ISoLA 2014, Corfu, Greece*, LNCS. Springer, Oct. 2014. In this proceedings.
9. G. Göri, E. B. Johnsen, R. Schlatte, and V. Stolz. Erlang-style error recovery for concurrent objects with cooperative scheduling. In T. Margaria and B. Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation, 6th International Symposium, ISoLA 2014, Corfu, Greece*, LNCS. Springer, Oct. 2014. In this proceedings.
10. I. lanese, M. Lienhardt, M. Bravetti, E. B. Johnsen, R. Schlatte, V. Stolz, and G. Zavattaro. Fault model design space for cooperative concurrency. In T. Margaria and B. Steffen, editors, *Leveraging Applications of Formal Methods, Verification*

and Validation, 6th International Symposium, ISoLA 2014, Corfu, Greece, LNCS. Springer, Oct. 2014. In this proceedings.

11. J. Woodcock, A. Cavalcanti, J. Fitzgerald, S. Foster, and P. G. Larsen. Contracts in CML. In T. Margaria and B. Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation, 6th International Symposium, ISoLA 2014, Corfu, Greece*, LNCS. Springer, Oct. 2014. In this proceedings.