CloudScale: Scalability Management for Cloud Systems

Gunnar Brataas*,**
Erlend Stav*
* SINTEF ICT, ** IDI, NTNU
Trondheim, Norway
{Gunnar.Brataas,
Erlend.Stav}@sintef.no

Sebastian Lehrig
Steffen Becker
Universität Paderborn
Paderborn, Germany
{Sebastian.Lehrig,
Steffen.Becker}
@uni-paderborn.de

Goran Kopčak
Darko Huljenic
Ericsson Nikola Tesla
Zagreb, Croatia
{Goran.Kopcak,
Darko.Huljenic}@ericsson.com

ABSTRACT

This work-in-progress paper introduces the EU FP7 STREP CloudScale. The contribution of this paper is an overall description of CloudScale's engineering approach for the design and evolution of scalable cloud applications and services. An Electronic Health Record (EHR) system serves as a motivation scenario. The overall CloudScale method describes how CloudScale will identify and gradually solve scalability problems in this existing applications. CloudScale will also enable the modelling of design alternatives and the analysis of their effect on scalability and cost. Best practices for scalability will further guide the design process. The CloudScale method is supported by three integrated tools and a scalability description modelling language. CloudScale will be validated by two case studies.

Categories and Subject Descriptors

D.2.2 [**Software**]: Programmer workbench; H.1.0 [**Information Systems**]: Models and Principles, General; D.2.11 [**Software**]: Patterns

Keywords

Scalability, Cloud, Software architecture, Provisioning, Domain specific modelling language, Tools

1. INTRODUCTION

Cloud providers theoretically offer their customers virtually unlimited infrastructure resources for their applications on an on-demand basis. However, scalability is not only determined by the available resources, but also by how the control and data flow of the application or service is designed and implemented. Implementations that do not consider their effects can either lead to low performance (under-provisioning, resulting in high response times or low throughput) or high costs (over-provisioning, caused by low utilisation of infrastructure resources).

ICPE'13, April 21–24, 2013, Prague, Czech Republic.

CloudScale [2] aims at providing an engineering approach for building scalable cloud applications and services with the following objectives:

1. Make cloud systems scalable by design so that they can optimally exploit the elasticity of the cloud, as well as maintaining and also improving scalability during system evolution. At the same time, a minimum amount of computational resources shall be used.
2. Enable model-driven analysis of scalability of basic and composed services in the cloud.
3. Ensure industrial relevance and uptake of the CloudScale results so that design for scalability becomes easier for cloud systems.

CloudScale's engineering approach for scalable applications and services will enable small and medium enterprises as well as large players to fully benefit from the cloud paradigm by building scalable and cost-efficient applications and services based on state-of-the-art cloud technology and software know-how. Furthermore, the engineering approach reduces risks and costs for companies entering the cloud market.

The consortium for the EU FP7 STREP CloudScale consists of five partners and is lead by the independent research institute SINTEF. SAP and Ericsson Nikola Tesla (ENT) are industrial partners, while the University of Paderborn (UPB) is an academic partner. XLAB is responsible for tool implementation and hosting activities.

The contribution of this work-in-progress paper is an overall description of the CloudScale approach to scalability management. The paper is structured as follows: First, we describe a motivating scenario in Section 2. CloudScale's Method, i.e., its engineering approach, is explored in Section 3. Validation and demonstrators are discussed in Section 4. Finally, Section 5 offers conclusions and an outlook on future activities.

2. SCENARIO

Ericsson Nikola Tesla (ENT) provides the Ericsson Healthcare Exchange (EHE) platform as part of its healthcare portfolio. The EHE platform is deployed on a national level in Croatia and several platform services are integrated in many healthcare provider institutions as shown in Figure 1. For example, the e-Prescription service is integrated in 2300+ general practitioner offices, 2500+ dentist offices, 192 paediatrician offices, 180 gynaecologist offices, and 1100+ pharmacies.

One of the main components of EHE is Electronic Health Records (EHR), which provide digitally stored health information for patients supporting care, education and research [7]. EHR stores information on medications, past medical history, immunisations, laboratory data, radiology reports etc [3]. The EHR system, as part of the EHE platform, needs to share data with healthcare providers, insurance institutions, government agencies, and patients.

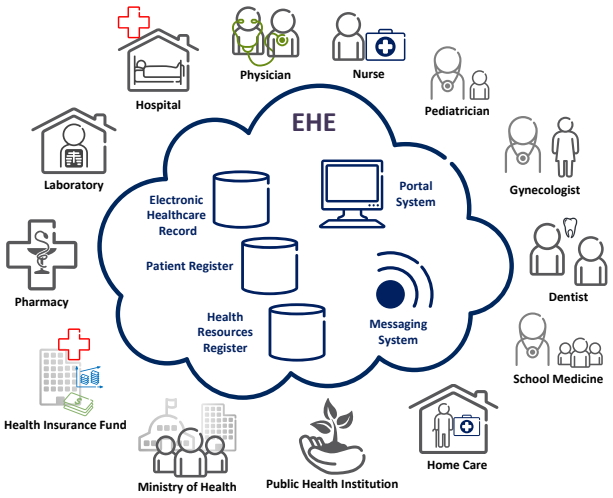

Figure 1: Ericsson Healthcare Exchange Platform.

The current non-cloud version of the EHR system is centralised and consists of the EHR database and a service layer developed above this database. Users can access EHR data either via direct database access or via the service layer. However, the main resource is always the database. Database mechanisms such as database clusters, connection pools, and database replication ensure system scalability and responsiveness up to some level that can be guaranteed by a large, but limited, system size. The resource demand per database user is non-linear.

A cloud-based solution is desired because of cost advantages, but also because of EHR requirements for scalability and elasticity. A cloud solution would also facilitate the provisioning of healthcare products and services to patients located in remote areas and others with limited access to quality medical services. EHR has the following characteristics which are particularly relevant for CloudScale's setting:

Variable load Since this solution is developed on a national level, the number of patients for which the healthcare records will be generated is known. Although the number of patients is predictable, the total number of EHR users can grow because of new services and integration of new institutions. Ericsson may also want to deploy the EHR technology in larger countries.

For specific annual events like regular flu or even major (not so frequent) events like bird flu, large earthquakes, or tsunami, there is a need for elastic system scale up. However, in holiday periods the system needs to scale down to reduce operational costs.

Constant system monitoring is necessary to be able to determine when and how to adjust the system size. Hence, a major advantage of a cloud based EHR solution, compared to the current non-cloud based solution, is system size elasticity.

Increasing work The size of the records will grow unpredictably, and some of EHR data records can be very large, e.g., treatment images, video files, sensor data (for remote patient monitoring).

Service Level Agreements (SLAs) Patient records may be required immediately. Users can be segmented in different hierarchy groups with different access priority, e.g., users from emergency healthcare institutions could have high priority.

Simple switching to a cloud infrastructure does not solve these problems without additional analysis and refactoring. Using CloudScale's approach, it should be possible to validate new requirements, evaluate system scalability, and provision an optimal amount of infrastructure resources. Now, it is very hard to predict the appropriate amount of system resources in a cost effective way, especially for custom defined user requirements.

In the process of switching from non-cloud EHR solution to the cloud infrastructure, Ericsson can use the CloudScale method and tools to monitor and optimise system scalability and also for predicting system size for a given workload. CloudScale can offer techniques analysing the existing solution, detecting architectural problems, and propose good architectural patterns for system evolution. When developing new services and identifying new customer requirements, design-time predictions could be made and this would also enable tuning of system specifications. In that way, EHR provider can forecast scalability properties and necessary system resources and have early operation cost predictions. During runtime, a constant monitoring processes can provide system elasticity to changes in workload.

3. METHOD

To illustrate how CloudScale aims at achieving the solution to the issues explained in Section 2, this section presents CloudScale's Method, its engineering approach for scalable cloud applications. We describe (1) the process steps as dictated by the Method, (2) main artefacts used in the Method (the scalability description language ScaleDL as well as (scalability) patterns and anti-patterns), and (3) the three tools Analyser, Extractor, and Spotter.

The proposed Method builds on an overall system lifecycle process illustrated in Figure 2 that focuses on scalability. As any other development process, it starts with requirements identification (Process (1)). During requirements identification, the main focus is on describing the load and work scaling path, i.e., the anticipated evolution of load and work on the system. The resource scaling path describes how our cloud platform can increase its amount of infrastructure resources, and a quality metric describe what is acceptable system quality to the users, e.g., a particular response time. For example, an emergency room using the EHR system could require to get one particular health record in less than one second. This process results in a requirement specification document.

Based on this requirement specification, the process of system construction and analysis starts (Process (2)) where we will use a model for specifying the system. We can do this in two ways:

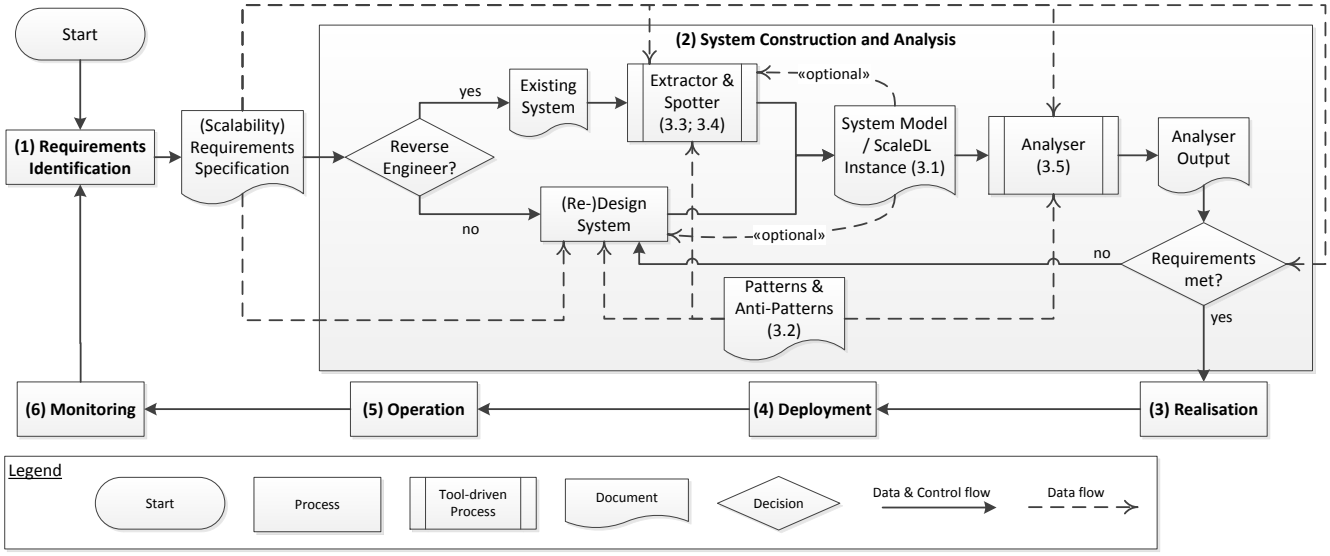

Figure 2: The overall CloudScale Method.

- *Reverse engineer* using an existing code base for creating an initial or adapting an existing system model. This process is driven by our tools Extractor and Spotter as described in Section 3.3 3.4, respectively.

- *(Re-)Designing* a system on the model level using our Analyser tool as described in Section 3.5. For specifying this model, we will use the ScaleDL language in Section 3.1.

We guide our design decisions along the requirements specification and support it with known patterns for good architectures regarding scalability in cloud environments. Section 3.2 provides details on patterns and anti-patterns.

The next step involves the analysis of the modelled system to check whether it meets the identified requirements. This process is driven by our Analyser tool as described in Section 3.5 and is repeated with different system alternatives until the requirements are met. The system construction and analysis process finally ends by reaching satisfactory results. We may also find that our requirements are infeasible and, therefore, have to relax them by reducing the complexity (and consequently work) of the services offered during high load.

After that, the system will be implemented (Process (3)), deployed (Process (4)), and put in operation in a cloud environment (Process (5)). For operation, it has well-defined resource requirements that enable cloud infrastructure provider to provision resources and fulfil load and work requirements of the deployed application reflecting the application evolution path.

Monitoring is another process (Process (6)) that is active during the system operation and enables control of system behaviour. Collecting measurements for performance parameters also belongs in this step. Based on the operational parameters and system quality metrics, monitoring control can require some changing system requirements and triggering the need to rerun our process cycle (evolution).

3.1 ScaleDL

ScaleDL will be a language for making precise models of the scalability properties of basic and composite cloud services. ScaleDL models will include structural elements of the system, parameters affecting scalability such as their resource usage, and elasticity information. Infrastructure resources will be described in a uniform manner, so that different configurations can be compared with each other. Among the resources to include in the model are processors, storage and network connections.

ScaleDL models can be created manually by the modeller based on knowledge and educated guessing, automatically by code analysis, measurements and predictions, or a combination of these. Ultimately, the scalability models can be used to determine resource scalability and cost scalability of the system, and help service providers to determine, e.g., what capacity the system will provide for a given cost. One of the design criteria for ScaleDL is that it must be useable in combination with scalability requirements specifications for the system, allowing the analysis of different scenarios and of fulfilment and consequences of the requirements.

ScaleDL will build on and extend existing languages and concepts where practical and appropriate. This includes the performance prediction language PCM of Palladio [5], the concept of system size from the scalability framework described in [6], and languages such as MARTE [4]. The design of ScaleDL will also be aligned with the ongoing development of the CloudML modelling language [1] which focuses on deployment models for cloud services.

3.2 Patterns and Anti-Patterns

Best practice scalability patterns show service providers how to build cloud systems with good scalability, while anti-patterns describe typical ways of making systems with poor scalability. For each anti-pattern, a recommended pattern is provided as well as a process for the transformation. Our EHR system can, for example, be suggested to be restructured according to a pattern that removes scalability issues with its database connection pool.

3.3 The Extractor

In case there is already existing code, the service provider can use the Extractor to extract an initial ScaleDL model to start from. This tool will enhance the reverse engineering tool Archimetrix [8] to take scalability parameters into account. The Extractor is also used during system evolution to keep the code and the scalability model in sync. In the EHR system, the Extractor would, e.g., find the data store component and its properties.

3.4 The Spotter

CloudScale's Spotter supports the service provider in spotting the places in a system that may cause scalability problems. It uses a ScaleDL model, AST's of code, monitoring data, and performance predictions to detect scalability anti-patterns. The Spotter can, e.g., be used to identify what needs to be changed to comply with new requirements and load/work plans. In the EHR system, the Spotter could, identify any non-linearites in the connection pooling that cause too many required connections when new users arrive.

3.5 The Analyser

Sooner or later, changing requirements or a changing work or load plan will trigger system redesign to still operate cost efficiently. The Analyser tool will make scalability predictions both of completely new designs or of redesigns based on existing systems. It will also analyse composed services. Using the Analyser, design alternatives can be analysed for their effect on scalability and cost. Internally, model transformations are employed to create scalability prediction models that build on established performance analysis approaches. We will base ourself on the open source tool set Palladio [5], which supports simulation as well as analytical solving.

Regarding our requirement that the EHR system should respond within one second, the Analyser may predict that it is capable of handling the increased load in case a bird flue occurs. Particularly, the Analyser takes the cloud systems' elasticity into account for its analysis.

4. VALIDATION AND DEMONSTRATORS

Demonstrators will show all the aspects of CloudScale in the form of prediction, analysis, metrics, measurements, anti-patterns, etc. They will provide a story line to support dissemination and make a platform for exploitation. The demonstrators will be completely open and realistic and will overcome any confidentiality issues of the industrial partners SAP and ENT in presentation of their internal systems.

The validation of CloudScale's showcases will collect evidence that (a) the CloudScale methods and tools spot the right bottlenecks, (b) our scalability predictions are sound, and (c) near optimal scalability improvements of a system are suggested. We also validate the feasibility of the CloudScale Method and report on the efforts needed to use it.

5. CONCLUSIONS

This paper has presented the CloudScale project at an overall level. The planned results of CloudScale are aimed at different types of people, organisations and roles, offering benefits to each.

- *End users* being satisfied also during peak loads because of improved scalability of the systems.
- For *developers* of software services, improved scalability management becomes a selling point. CloudScale tools help developers to make sensible decisions about which parts of the system which require extra care.
- *System architects* are able to understand and predict the scalability of services resulting from compositions.
- *Service providers* are able to make timely decisions about purchase or deployment of more hardware in order to prevent scalability bottlenecks before they show up. They are also able to plan reduction in non-essential features to retain core functionality during periods of extreme demand.
- *IaaS (Infrastructure as a Service)* providers will optimise their business due to more effective use of resources by their customers, thanks to improved scalability. Moreover, they are able to serve more customers with the same hardware through better management of scalability of their own systems. Thereby, they can operate with a smaller safety margin and greater profit.

The CloudScale project started in October 2012 and will continue until September 2015. The first versions of the results are planned in October 2013. This includes a refined Method, ScaleDL, the three tools Analyser, Spotter, and Extractor, as well as patterns and anti-patterns.

6. ACKNOWLEDGEMENT

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant no 317704 (CloudScale).

7. REFERENCES

[1] Cloud modelling language (CloudML). `www.cloudml.org`.
[2] CloudScale Project. `www.cloudscale-project.eu`.
[3] Healthcare Information and Management Systems. `www.himss.org`.
[4] Modelling and Analysis of Real-Time and Embedded Systems (MARTE). `www.omgmarte.org`.
[5] S. Becker, H. Koziolek, and R. Reussner. The palladio component model for model-driven performance prediction. *Journal of Systems and Software*, 82(1):3 – 22, 2009.
[6] G. Brataas, P. H. Hughes, J.-A. Fagerli, and O. C. Landmark. Exploring Architectural Scalability. *Software Engineering Notes*, 29(1):125 – 129, 2004.
[7] I. Iakovidis. Towards personal health record: current situation, obstacles and trends in implementation of electronic healthcare record in Europe. *International Journal of Medical Informatics*, 52(1-3):105 – 115, 1998.
[8] M. Platenius, M. von Detten, and S. Becker. Archimetrix: Improved software architecture recovery in the presence of design deficiencies. In *Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on*, pages 255 –264, march 2012.