Multi-tenancy in Software as a Service Applications

Eveliina Pakarinen
University of Helsinki
Helsinki, Finland

Abstract—**Multi-tenancy is a high level architectural pattern which can be used in cloud computing environment when offering applications using Software as a Service business model. In multi-tenancy pattern service provider hosts a single instance of an application on his or her infrastructure and this application is accessed by multiple tenants who share the resources of the infrastructure. The use of multi-tenancy pattern brings multiple benefits both to the service provider and to the customers but there are also some complexities that affect the use of multi-tenancy. Although multi-tenancy is a popular paradigm it is still a relatively new topic in scientific literature and the research domain of multi-tenancy is not yet mature which can be seen from the lack of industrial experience reports on multi-tenancy. In this paper the key concepts of multi-tenancy in SaaS applications are presented and some concerns that need to be taken into account when developing a multi-tenant SaaS application are discussed. A couple of recently proposed example frameworks for developing multi-tenant applications are also presented in order to describe the current situation in multi-tenancy research.**

Keywords: multi-tenancy, Software as a Service, architectural pattern

I. Introduction

Software as a Service (SaaS) is a software delivery and business model where an application is provided as an on-demand service for multiple users through Internet [1]. In SaaS business model the service provider maintains the application and offers the software as a service to the customers [2]. By using software offered by a third party companies can use various IT services without maintaining or purchasing their own IT infrastructure [2].

Multi-tenancy is a high level architectural pattern which can be used when offering applications as Software as a Service in cloud computing environment [3]. In multi-tenancy pattern the service provider hosts a single instance of the software product on his or her infrastructure and multiple customers, so called tenants, access the same instance of the software [2]. A tenant is the organizational entity which rents a multi-tenant SaaS solution [2]. A tenant groups typically multiple users of the same organization and these users are the stakeholders in the organization.

There are multiple benefits for the service provider when using multi-tenant architecture pattern when implementing SaaS applications. The first benefit is that the application deployment becomes easier because only one application instance has to be deployed [2]. In multi-tenant model multiple customers access the same software instance and they do not need own dedicated instance of the software. This means that the customers share the same hardware resources when using multi-tenant SaaS application [4]. That increases and improves the utilization rate of the hardware which is the second benefit of the multi-tenant model [2].

These two benefits reduce the software delivery costs for the service provider which can help improve the profit margin [5]. The reduced delivery costs enable that software provider can offer the service to the customers at lower service subscription costs [5]. That makes multi-tenant applications interesting for customers in the small and medium enterprise segment [2].

In addition to the benefits of the multi-tenancy there are also some complexities that come with the multi-tenancy. Challenges can arise in the application development, deployment and management phases [5]. Challenges that arise are for example application performance, scalability, security, zero-downtime and maintenance [2].

Although multi-tenancy is a popular paradigm it is still a relatively new topic in scientific literature [3]. The term multi-tenancy was explicitly mentioned for the first time in a scientific paper in year 2006. Since then many definitions for multi-tenancy have been proposed [3]. Also many solutions related to multi-tenancy have been proposed in the multi-tenancy research over the years but there has been very few industrial reports about experiences on multi-tenancy [3]. This indicates that the research domain of multi-tenancy is not yet mature and that the solutions have not yet been implemented or evaluated. The high amount of proposals and the low amount of industrial reports can also indicate that there is a lack of cooperation between industry and academia in this domain [3].

In this paper the key concepts of multi-tenancy in SaaS applications are presented and some concerns that need to be taken into account when developing a multi-tenant SaaS application are discussed. A couple of recently proposed example frameworks for developing a multi-tenant applications are also presented in order to describe the current situation in multi-tenancy research.

This paper is organized as follows. In Section 2 the research methods for data collection for this paper are introduced and the research questions are presented. In section 3 an introduction to multi-tenancy and SaaS is given. In section 4 an example architecture framework for multi-tenant architecture is presented. In section 5 the answers to the research questions are discussed. A conclusion is presented in section 6.

II. Research methods

This seminar work is based on academic papers published in various conferences and journals. The search for these papers was mainly done in the ACM Digital Library and in the IEEE Xplore Digital Library. Initial search criteria for papers were that the paper was published recently between years from 2014 to 2016 and that the paper had something to do with multi-tenancy and architecture or multi-tenancy and SaaS applications or generally mentioned multi-tenancy. After finding a couple of papers the references in them were used to find more papers related to the theme discussed in the paper. Web of Science was also used to find papers that had cited the papers found during the search in digital libraries.

The initial goal for data collection was to find as many papers as possible. After that the first thing to do was to filter out papers that were not suitable as sources for this seminar work. Characteristics for not suitable papers were for example that the paper was published in a too small conference or unknown journal or that it did not discuss multi-tenancy in application level in SaaS applications or that multi-tenancy was not discussed from architectural perspective.

After that the remaining papers were organized to four groups based on the abstract, introduction and conclusion parts of the papers. The groups in which the papers were organized were:

1) not so relevant papers for this seminar work
2) important papers for this seminar work
3) papers that propose some multi-tenant pattern or architecture
4) supporting papers with relevant background information.

The papers that belonged to group 1 were left out and the papers that belonged to groups 2, 3 and 4 were studied further. The references that are used in this seminar work are based on the papers that belonged to groups 2, 3 and 4.

The main research questions for this seminar work are: What is multi-tenancy and what does it have to do with SaaS applications? What kind of architecture frameworks have been proposed for multi-tenant applications? What needs to be taken into consideration when developing multi-tenant applications? This seminar paper also tries to provide a clear description of multi-tenancy and multi-tenant SaaS applications.

III. Introduction to multi-tenancy and SaaS (software/architecture/patterns)

Multi-tenancy is a high level architectural pattern which can be used to share computing resources when offering software products as Software as a Service [3]. In multi-tenancy a single instance of an application is hosted on service providers infrastructure and this single instance is accessed by multiple tenants and can be customized according to the requirements of different tenants [3]. Multi-tenancy has evolved from previous information technology paradigms like time-sharing, application service provider (ASP) model and multi-user model [3]. Multi-tenancy was explicitly mentioned in scientific literature for the first time in the domain of software and hardware systems in year 2006 [3].

In academic literature the definition of multi-tenancy has been varying and there has been differences in the interpretation of multi-tenancy among academics in academia [3]. In order to chart and bridge the varying definitions of multi-tenancy and to provide an overview of the multi-tenancy domain Kabbedijk et al. [3] performed a structural search in academic literature and blog posts. They propose a comprehensive definition for multi-tenancy which is based on the definitions of multi-tenancy presented in academic literature [3]. Their definition of multi-tenancy is: Multi-tenancy is a property of a system where multiple customers, so-called tenants, transparently share the systems resources, such as services, applications, databases, or hardware, with the aim of lowering costs, while still being able to exclusively configure the system to the needs of the tenant [3].

In Figure 1 different system levels are illustrated in different variants of multi-tenancy and in single-tenancy for two tenants. In Figure 1 the system levels marked with yellow background color represent application and software levels of the system. The infrastructure levels of the system are marked with purple background color and the database levels are marked with green background color. The levels of the system that are influenced by software level multi-tenancy are marked with solid borders and bold font style. Those levels are application instance, application server, database schema, database and database server [3]. The levels that are not influenced by software level multi-tenancy are middleware, operating system, virtual machine and hardware and those levels are marked with dashed line borders and plain font style in Figure 1.

In native or pure multi-tenancy [3], [5] the tenants share the same application instance and database tables which are hosted on a shared infrastructure. Native multi-tenancy variant is used to support a large number of tenants and the number is usually in the hundreds or thousands of tenants [5].

In the semi-multi-tenancy variants of multi-tenancy the tenants share the same application instance but the level of sharing on the database levels of the system vary. The two variants of semi-multi-tenancy are shared application, shared database and separate table -variant and shared application and separate database -variant [2]. When a high number of tenants are placed on the same server and one of the semi-multi-tenancy variants of multi-tenancy is used it can cause performance problems on the server [2]. These performance problems are caused by the expensive operation of loading a database or table in memory when a tenant accesses the application. When comparing this to native multi-tenancy native multi-tenancy enables placing more tenants on the same the server because the shared database table must be loaded only once to memory [2].

In multiple instances multi-tenancy tenants no longer share the same application instance. Instead of sharing an application instance each tenant has its own dedicated application instance over a shared middleware server, operating system or hardware [5]. Two variants of multiple instances multi-tenancy are presented in the Figure 1. Multiple instances multi-tenancy variants scale differently than native multi-tenancy when comparing the number of tenants which multiple instances variants or native multi-tenancy can support [5]. Multiple instances multi-tenancy variants can support several up to dozens of tenants while native multi-tenancy can support hundreds or thousands of tenants [5].

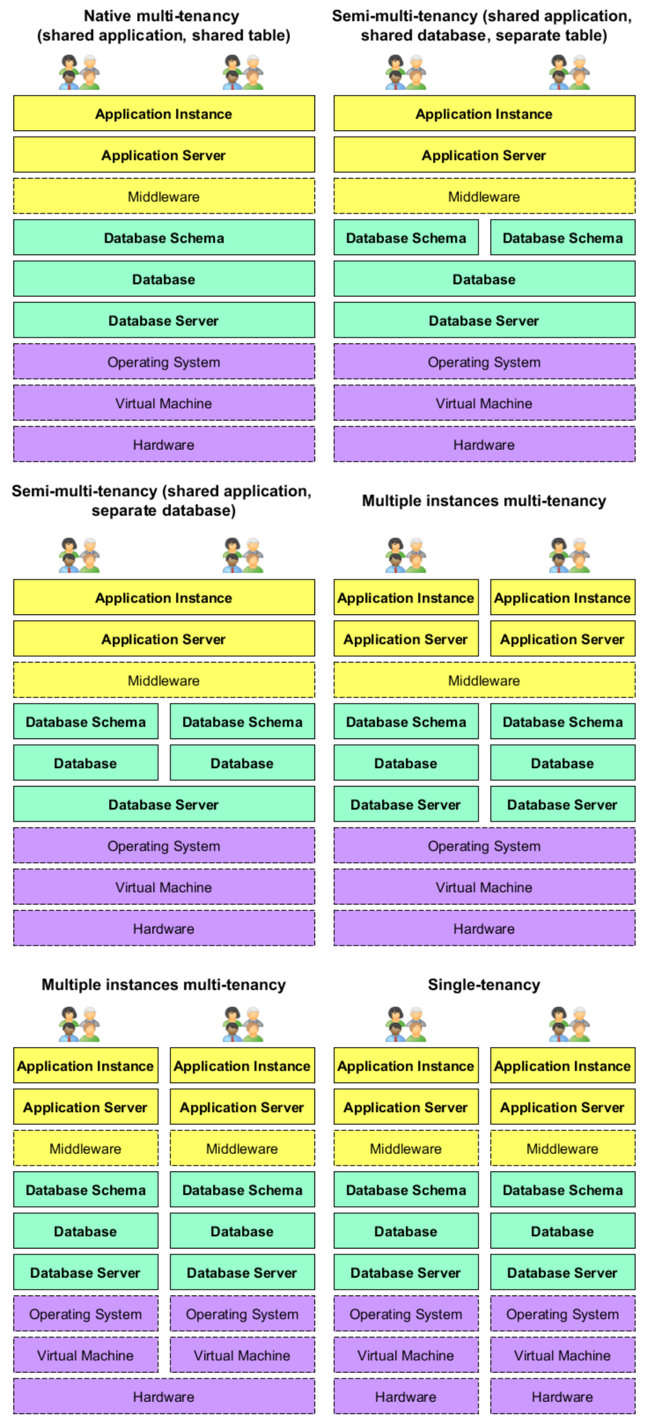

Fig. 1. Illustration of different system levels in different variants of multi-tenancy and in single-tenancy.

As a comparison point to different multi-tenancy variants a single-tenancy approach is also presented in Figure 1. In traditional single-tenant approach each tenant has own dedicated server and own customized application instance which they use [2]. This means that single-tenant application may have many separate running instances which all can be different from each other [2]. A traditional single-tenancy scenario from the early 90s was that companies moved their hardware and applications from their premises to data centers where the applications were hosted without any hardware or software sharing [5].

In single-tenancy server utilization can be low especially when applications are used by customers in the small and medium enterprise (SME) segment of the market [2]. In that situation server utilization can be improved by placing several tenants on the same server [2]. Different multi-tenancy variants from multiple instances multi-tenancy to native multi-tenancy can be used to place several tenants on the same server which improves the utilization of the servers. Through higher utilization of the servers the total amount of hardware required to serve the tenants is lower than when using single-tenancy where each tenant has its own dedicated server. The result of requiring lower amount of hardware because of higher server utilization in multi-tenancy is that the overall costs of the application will be lower [2].

IV. Results: Evaluation/validation/review of a proposed architecture pattern/framework for multi-tenant SaaS application

(or maybe comparison to some other framework? How has it evolved through years?)

Answers to the research questions (should be here?)

New results

Neutral analysis based on the data. New ideas/new things/ new viewpoints based on the material my unique way to combine the materials to something new?

V. Discussion

How well the research questions were answered?

Discuss my answers to research questions

Limitations that affect the validity of the results

Related work or comparison to other work (maybe)

VI. Conclusion

Main points, message

Probably some mention about future work or what should be researched next

References

[1] S. Kang, S. Kang, and S. Hur, “A design of the conceptual architecture for a multitenant saas application platform,” in *2011 First ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering*, May 2011, pp. 462–467.

[2] C.-P. Bezemer and A. Zaidman, “Multi-tenant saas applications: Maintenance dream or nightmare?” in *Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE)*, ser. IWPSE-EVOL ’10. New York, NY, USA: ACM, 2010, pp. 88–92. [Online]. Available: http://doi.acm.org.libproxy.helsinki.fi/10.1145/1862372.1862393

[3] J. Kabbedijk, C.-P. Bezemer, S. Jansen, and A. Zaidman, "Defining multi-tenancy: A systematic mapping study on the academic and the industrial perspective," *Journal of Systems and Software*, vol. 100, pp. 139 – 148, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121214002313

[4] S. Pal, A. K. Mandal, and A. Sarkar, "Application multi-tenancy for software as a service," *SIGSOFT Softw. Eng. Notes*, vol. 40, no. 2, pp. 1–8, Apr. 2015. [Online]. Available: http://doi.acm.org.libproxy.helsinki.fi/10.1145/2735399.2735412

[5] C. J. Guo, W. Sun, Y. Huang, Z. H. Wang, and B. Gao, "A framework for native multi-tenancy application development and management," in *The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007)*, July 2007, pp. 551–558.