ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A Review of Software-as-a-Service (SaaS): Features and Multi-Tenant Applications

Trilochan Kumar Patel*, Menka Patel
* IT Dept., Kirodimal Institute of Technology, Raigarh(C.G.), India
English Dept., Govt. H. S. School Kauhakuda, Pithora, Mahasamund(C.G.), India

Abstract

Multi-tenancy is a relatively new software architecture principle in the realm of the Software as a Service (SaaS) business model. It allows to make full use of the economy of scale, as multiple customers –"tenants" – share the same application and database instance. All the while, the tenants enjoy a highly configurable application, making it appear that the application is deployed on a dedicated server. The major benefits of multi-tenancy are increased utilization of hardware resources and improved ease of maintenance, in particular on the deployment side. These benefits should result in lower overall application costs, making the technology attractive for service providers targeting small and medium enterprises (SME). However, as this paper advocates, a wrong architectural choice might entail that multitenancy becomes a maintenance nightmare.

Keywords: Multi-Tenancy, Software as service, Business Model, Small and Medium Enterprises (SME).

Introduction

SaaS (or Software As A Service) is a hot topic in the hosting industry. However, SaaS is not a new concept and in one form or another it has existed for many years. Recently SaaS became especially popular. Maybe this is mainly because of the recession and the need to cut costs, rather than because of the the advantages of SaaS itself. Anyway, now SaaS is drawing more attention and is frequently regarded as the way to drastically cut costs with no sacrifice to quality.

SaaS is a form of cloud computing. In fact, it is maybe the most popular form of cloud computing because it is targeted at the general public, rather than on the IT community itself. The idea of SaaS is simple: you subscribe to a SaaS service provider vendor, who installs and configures the software applications you need and you just use these applications. Basically, you rent software rather than buy it and you also spare yourself the hassle to install and deploy these applications.

For anybody, who is not familiar with the process of purchasing, installing, and deploying software, SaaS might look a useless service. However, anybody, who has had the pleasure to choose a software application, pay a hefty price for it, spend months on installation, deployment, configuration, customizations, etc., SaaS looks like the answer to their prayers. Well, it is not correct to say that SaaS is only gold - not at all, SaaS also has its disadvantages - but as a whole, the SaaS model is efficient in many ways.

What is SaaS ?

Software as a service (or SaaS) is a way of delivering applications over the Internet—as a service. Instead of installing and maintaining software, you simply access it via the Internet, freeing yourself from complex software and hardware management. SaaS applications are sometimes called Web-based software, on-demand software, or hosted software. Whatever the name, SaaS applications run on a SaaS provider's servers. The provider manages access to the application, including security, availability, and performance.

SaaS features

Related work

ISO 9126 is an international standard for the evaluation of product quality [01]. This standard provides three aspects for evaluating software products; internal quality, external quality, and quality in use. And, there are sixteen characteristics for three types of qualities. However, this standard focuses on evaluating quality of conventional products. Hence, it is required that the standard is customized and extended to evaluate the quality of SaaS. Jureta's work proposes a quality model, called QVDP, to measure the quality of Service-Oriented System [02]. This model consists of four sub models; quality

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

characteristic, characteristic value, quality dependency, quality Priority. These represent dependencies and priorities between qualities dimensions and quality characteristics. However, this work considered services-oriented application as a target of quality model and identifies issues related to them at conceptual level. Kim's work defines a model for web services quality management and quality factors in the process of developing and using web services [03]. This work suggests six quality factors and their several sub factors. Also, it provides metrics to measure quality factors. Hence, it is required that this model is customized and extended to evaluate the quality of SaaS. Most of current works are not for SaaS but for certain targets such as a conventional software or SOA based system. Due to the situation, it is hard to evaluate quality of SaaS and judge which SaaS is good. Therefore, our work provides a quality model to evaluate SaaS.

Features of saas application performance monitoring tools

Software-as-a-Service (SaaS) is relatively new to most markets and companies, and third-party backups for these cloud applications are newer still. What are the must-have features for a SaaS backup system? Here are the 16 most-requested features -

1. **Backup Historical and Progressive Data**

A backup solution should protect all the data in your system, not just the data that's added after you install the backup. If you need to restore data that has been unaltered for three years – which is precisely the sort of historical content that is prone to corruption or accidental deletion – your backup system should have a copy of it on hand.

2. **Data Versioning (Incremental Backups)**

Backing up just the most recent version of SaaS data means that – if you don't catch an error before your backup archive updates – you have two copies of corrupted data. Data versioning means you have multiple copies of the same data elements, each captured at regular intervals, allowing you to roll back to whichever state has the most accurate or necessary information.

3. **Local Export Options**

Data trapped in your cloud application should not be equally trapped in your cloud application backup. Your SaaS backup provider should offer local download and export options so you can keep local copies of any items (or even accounts) you deem fit.

4. **Scheduled and User-Initiated Backups**

Regular, scheduled, automated backups ensure that no critical data is omitted from your archives simply because an administrator forgot to trigger a backup. User-initiated backups ensure that, following a critical update to live SaaS data, the backup archive can be immediately updated to ensure this data is protected. A competent backup system should offer features, rather than simply one or the other.

5. **Proactive Status Updates and Error Notification**

Backup administrators shouldn't have to log into a backup system – let alone individual backup accounts – to learn whether a backup process has been successful. The backup application should proactively alert admins to backup failures and, ideally, allow an admin to diagnose and correct the problem as soon as possible from a central interface.

6. **Support for Your Recovery-Time Objectives (RTO)**

It's not enough for a cloud application backup to restore data; it must restore data fast enough that your business isn't significantly harmed by data loss. How long does it take to restore one item, one account, or the complete data archive? Know the answer before you deploy your SaaS backup solution.

7. **Restore for Individual Items**

Rarely is all the data corrupted or erased from a cloud application; a typical data loss scenario involves only a handful of missing or damaged items. Your SaaS backup solution should allow you to restore just those items – the mail your user accidentally erased, or the single table your database dropped – rather than deal with reinserting a complete copy of your entire account (or archive) back into your cloud application.

8. **System-Wide Search**

Most restoration tasks involve single items, so your SaaS backup solution should make it easy to find those individual items within your archives – and that requires system-wide search. Manually browsing chronological archives can significantly slow down restore efforts, and search is a must have shortcut to ensure your Restore-Time Objectives are met.

9. **Comprehensive Data Backup (AKA "Complete Suite" Backup)**

Many SaaS backup products only backup a portion of the data in your cloud application, often leaving out certain feature sets (backing up text but not images, documents but not emails) or ignoring key metadata

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

(emails but not attachments; documents but not their tags and access control lists, etc.). Your cloud application backup should protect every data type necessary to keep your SaaS solution running with full data integrity.

10. Centralized Account Management

Administrators should be able to view backups and archives for all accounts through a single interface, so that as your business grows and you add new employees, account administration stays fast and efficient.

11. Robust Permission Controls

Administrators should be able to monitor and control what features their users can enable, disable or configure. Backups do you no good if end users can prevent or delete them without administrator knowledge or consent.

12. Streamlined, Versatile Onboarding

A backup solution should allow administrators to quickly opt-in which user accounts to include in the backup archives. Mandatory backups for all accounts are unacceptable, as is a tedious, manual selection process. The SaaS backup solution should also allow for new SaaS application users to automatically be backed up.

13. Documented Security Procedures

A backup provider should offer documented security procedures for the transfer and protection of your data. It's not enough to claim to be "secure;" your SaaS backup provider should be prepared to give reasonable specifics – like level of encryption of data at rest, and which data transactions occur over SSL – to assure the safety of your data.

14. Documented Support Options

A cloud application is only as good as its technical support, and this goes doubly so for SaaS backup services, as you'll be relying on your backups to function during times of need. Your SaaS backup solution should have a clear method for contacting technical support and self-service support options (like FAQs and help forums) so you can work towards solving problems on your own,
without waiting on a response from the service provider. It's not either-or, it's both.

15. Documented Service Level Agreement

Just as you wouldn't purchase a SaaS product or cloud application without a documented Service Level Agreement, the system backing up your cloud apps should have an SLA. Specifically, your backup system should lay out explicit uptime guarantees and the compensation provided if those guarantees aren't met.

16. Flexible Billing

Your SaaS backup solution should offer as much contract flexibility as the service it's backing up. It should fit into your existing buying and budget cycle, not force another one on you.

Multi-tenancy

Multi-tenancy is an organizational approach for SaaS applications. Although SaaS is primarily perceived as a business model, its introduction has lead to numerous interesting problems and research in software engineering. Despite the growing body of research in this area, multi-tenancy is still relatively unexplored, despite the fact the concept of multitenancy first came to light around 2005.

While a number of definitions of a multi-tenant application exist [04,05], they remain quite vague. Therefore, we define a multi-tenant application as the following:

Definition 1. A **multi-tenant** application lets customers (**tenants**) share the same hardware resources, by offering them one shared application and database instance, while allowing them to configure the application to fit their needs as if it runs on a dedicated environment.

Definition 2. A **tenant** is the organizational entity which rents a multi-tenant SaaS solution. Typically, a tenant groups a number of users, which are the stakeholders in the organization.

These definitions focus on what we believe to be the key aspects of multi-tenancy:
1. The ability of the application to share hardware resources.
2. The offering of a high degree of configurability of the software.
3. The architectural approach in which the tenants (or users) make use of a single application and database instance.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

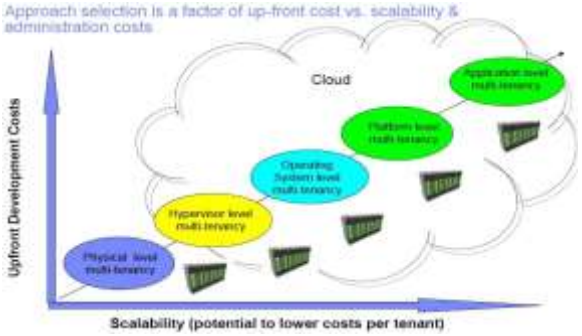

Figure 1 : Type of Level multi-tenancy

Type I - Physical Level multi-tenancy
Characteristics

- Better performance than virtualization because using native hardware.
- Runs non multi-tenant applications w/o changes.
- Poorest scalability (number of tenants per server = 1).
- Highest operational costs.
- Provisioning cannot be fully automated.

When to use this model

Generally hard to justify use of this model except if customers require dedicated servers because of regulation / standards.

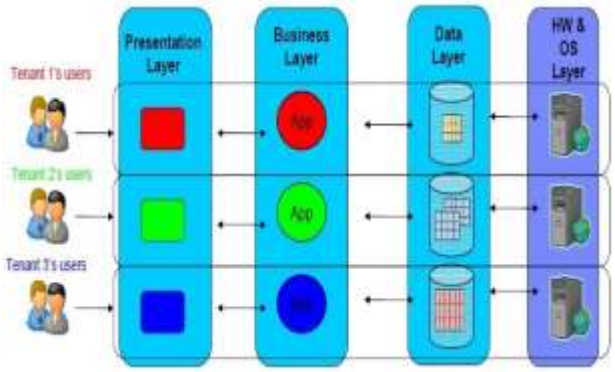

Figure 2 : Physical Level multi-tenancy

Application Service providers host each tenant's application dedicated hardware, middleware and operating system.

Type II - Hypervisor Level multi-tenancy
Characteristics

- Runs non multi-tenant applications w/o changes.
- Drop off in performance from Type I because applications have 2 Operating Systems to traverse to get to hardware.
- Better scalability than physical level (number of tenants per server > 1).
- Operational costs on par with Type I minus TCO of dedicated hardware.
- Provisioning can be fully automated.

When to use this model

• Application is not multi-tenancy aware and need to GTM quickly.

• Unknown demand for SaaS model and/or lack of resources to modify application.

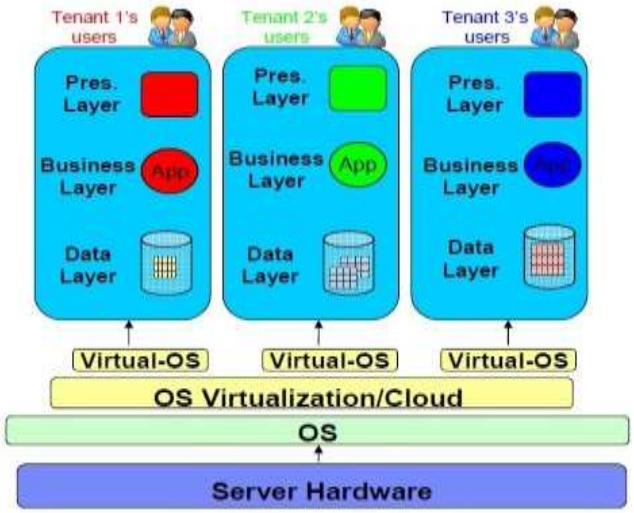

Figure 3 : Hypervisor Level multi-tenancy

Type III - Operating System Level multi-tenancy
Characteristics

- Runs non multi-tenant applications w/o changes.
- Can improve software license costs over Types I and II if middleware supports.
- multiple instances (e.g. database, application server) with a single license.
- Smaller footprint than Types I or II.
- Provisioning can be fully automated.

When to use this model

• Application is not multi-tenancy aware and need to GTM quickly.

• Unknown demand for SaaS model and/or lack of resources to modify application.

• Middleware supports multiple instances on single OS.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

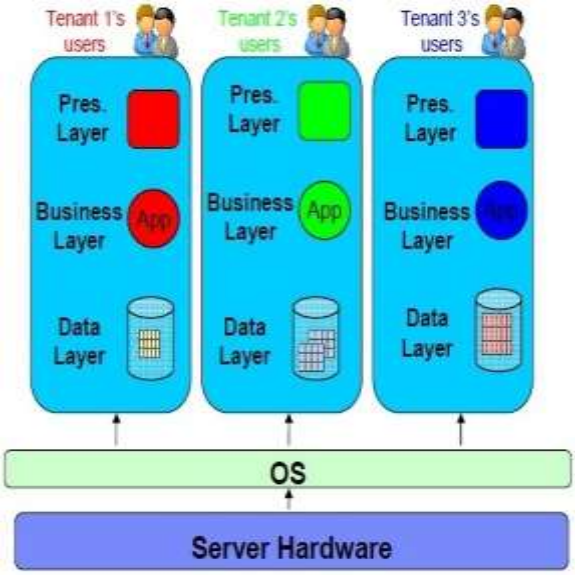

Figure 4 : Operating System Level multi-tenancy

Type IV – Platform Level multi-tenancy
Characteristics

- Cannot run non multi-tenant applications w/o changes.
- Can improve software license costs over Types I, II.
- Smaller footprint than Types I, II or III.
- Provisioning can be fully automated.

When to use this model

• New applications or resources exist to modify existing non-multi-tenant applications

• Market for type of application is extremely competitive driving prices down

• Middleware addresses technical challenges of true multi-tenant applications.

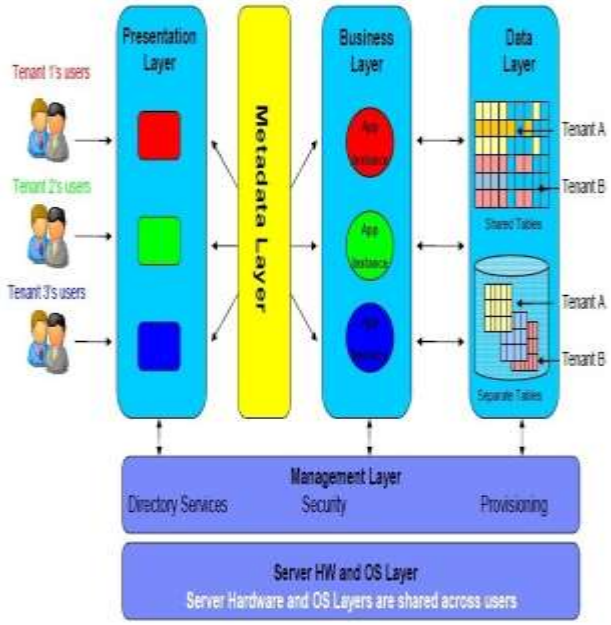

Figure 5 : Platform Level multi-tenancy

Type V – Application Level multi-tenancy
Characteristics

- Cannot run non multi-tenant applications w/o changes.
- Smaller footprint than Types I, II, III or IV.
- Provisioning can be fully automated.
- Mediator allows for advanced features like dynamic QoS adjustments.

When to use this model

• New applications or resources exist to modify existing non-multi-tenant applications.

• Market for type of application is extremely competitive driving prices down.

• Spikes in demand and/or SLA's require dynamic load balancing.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

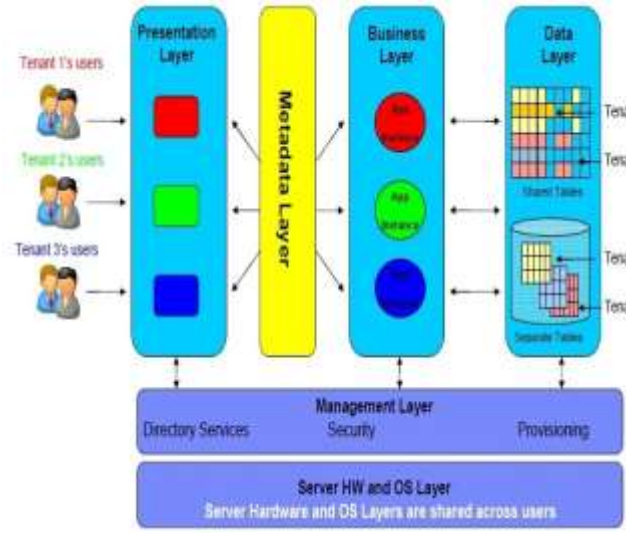

Figure 6 : Application Level multi-tenancy

Conclusion

Five major approaches for enabling multi-tenancy in Software-as-a-Service
1. Physical level
2. Hypervisor level
3. Operating System level
4. Platform level
5. Application level

IBM software, systems and services can help to build and deploy SaaS applications.

The SaaS blueprints show how to address many of the technical challenges in developing multi-tenant applications. this article is by no means the last word in single-instance, multi-tenant data architecture.

References

1. Software Engineering – Product Quality – Part 1 :Quality Model. ISO/IEC 9126-1, June, 2001
2. Jureta, I., Herssens, C., and Faulkner, S., "Acomprehensive quality model for service-orientedsystems," *Software Quality Journal*, to be published.
3. Kim, E. and Lee, Y., *Quality Model for Web Services*,Working Draft, OASIS, September 2005.
4. Zhi Hu Wang, Chang Jie Guo, Bo Gao, Wei Sun, Zhen Zhang, and Wen Hao An. A study and performance evaluation of the multi-tenant data tier design patterns for service oriented computing. In *Proc. of the Int. Conf. on e-Business Engineering (ICEBE)*, pages 94–101. IEEE, 2008.
5. Bob Warfield. Multitenancy can have a 16:1 cost advantage over single-tenant. http://smoothspan.wordpress.com/2007/10/28/ multitenancy-can-have-a-161-cost - advantage-over-single-tenant/ (last visited on June 2nd, 2010), October 2007.
6. https://www.ibm.com/developerworks/mydeveloperworks/groups/service/html /communityview?communityUuid=82fea6f2-2b51-447c-a118-88711258a502
7. IBM (2007b) *SaaS*, available at http://www-304.ibm.com/jct09002c/isv/marketing/saas/index.html.