Gartner Research

Publication Date: 27 June 2011 ID Number: G00213940

Understanding Tenancy: Salesforce.com Versus Google.com

Yefim V. Natis, Eric Knipp

Understanding the values and costs of multitenancy will help organizations select the appropriate approach to cloud computing. CIOs, CTOs, IT planners and software architects should seek insights regarding multitenancy in the technologies they evaluate for private and public cloud use.

Key Findings

- Tenant-enterprise multitenancy is the model in which the tenants are multiple instances of the same application, executing on behalf of isolated user organizations. This model is associated with software-as-a-service (SaaS) offerings that are used by multiple isolated customers in a shared physical environment. (The degree of sharing will vary.)
- Tenant-application multitenancy is the model in which the tenants are different, logically isolated applications running in a shared physical environment. (Again, the degree of sharing will vary.) This model is associated with the private cloud, where one physical environment hosts multiple applications competing for shared resources. It is also found with other cloud environments, especially those targeting multiple applications to individual consumers, rather than enterprises.
- "Gartner Reference Architecture for Multitenancy" describes seven patterns for implementing multitenancy. They differ in the degree of resource sharing among tenants, in the degree of backward-compatibility to precloud programming models, and in the degree of productivity and the speed of elastic scalability.
- Lack of multitenancy dramatically limits the benefits of off-premises software services, including infrastructure as a service (IaaS), platform as a service (PaaS) or SaaS. Minimal multitenancy (shared hardware) delivers minimal benefits of coarse elasticity.
- Advanced forms of multitenancy (shared container and shared everything) offer responsive, fine-grained elasticity that enables the provider or tenant to fine-tune and optimize the services more effectively, and improve the efficiency of the total resource utilization.

Recommendations

- Understand the trade-offs of the models of multitenancy and apply this understanding to the selection of cloud services (IaaS, PaaS or SaaS). The presence of multitenancy and its depth are good predictors of the long-term agility and efficiency of a cloud platform.

- Avoid multitenant platforms for projects that demand maximum control of the underlying environment and are not sensitive to the costs and agility of the application, but be prepared to develop your own support of multitenancy.
- Most projects that are aimed at preserving precloud engineering skills and precloud software will have to settle on less-advanced, but backward-compatible forms of multitenancy (such as shared hardware).
- Software designers, developers and architects who are willing to learn new models of programming, will have access to the benefits of advanced forms of multitenancy (such as shared everything).

TABLE OF CONTENTS

Analysis 4
- Multitenancy 4
- Sharing Versus Colocation 5
- The Two Main Types of Tenancy 6
  - Tenant-Enterprise 6
  - Tenant-Application 6
- Combining the Tenant Types in a Multitenant Environment 7
- Nested Multitenancy 7
- Is an Individual User a Tenant? 7
- Who Benefits From Multitenancy? 8
- The Bottom Line 9

Recommended Reading 9

Gartner

ANALYSIS

Elasticity of cloud computing is achieved by allocating computing resources to tenants on-demand and reassigning the shared resources from one tenant to another in response to changes in demand. Without tenants, there's no one to share resources, so there's no multitenancy; without multitenancy, there's no elasticity.

So what constitutes a tenant? Customers of salesforce CRM (organizations that sign up to use the application offered as a shared service) are universally understood as tenants, but what about the tenants of the Google.com site? What about a website such as nationalgeographic.com, for example? Do these sites also have tenants and, therefore, the opportunity for multitenancy and elasticity and, therefore, a claim to be cloud services? Are the many applications running in your own organization tenants of your data center?

A tenant in cloud computing is an entity for which a cloud platform tracks and optimizes the computing resources, while doing so discriminately for multiple entities (tenants), all of which are competing for the shared resources. In software terms, a tenant is an instance of an application that draws its essential computing resources from its underlying platform in the context of other application instances operating in the same environment. The underlying resources are shared by multiple application instances, and are managed on behalf of all of them. A policy-driven sharing of the computing resources among application instances (tenants) delivers the elasticity of the overall environment and makes its application services into cloud services.

Multitenancy

Tenants are *logically isolated* application instances that are executing in a *physically shared* environment (the degree of sharing varies). The origins of multitenancy go back to the mainframe time-sharing business in the 1960s. The new incarnation of the principles of multitenancy emerge in a much different computing environment and require largely different technical solutions, although some of the fundamentals still apply.

In a multitenant environment, each tenant operates as if the entire computing environment is its exclusive resource. Tenants are logically isolated, while they physically share some of the computing resources. How much of the computing platform is shared and how much of it is allocated exclusively to a tenant differentiates the various models of multitenancy (see "Gartner Reference Architecture for Multitenancy").

The isolated tenants must be completely isolated. The more physical resources are shared, the harder it is to retain complete isolation. Failure to isolate the tenants is unacceptable, and can take the offending offering off the market. Often, vendors choose to reduce sharing to avoid a potential loss of isolation.

In a multitenant environment, each tenant operates in a logically isolated, but physically shared environment, with the following characteristics:

- Tenant process isolation — including balanced performance characteristics, privacy of state and error isolation
- Tenant data isolation — including privacy and integrity
- Tenant customization
- Tenant security policies

- Tenant elasticity policies — including limits of automatic allocation, thresholds and increments of change
- Tenant SLA policies — including availability and response time
- Tenant resource-use tracking
- Optional tenant billing in proportion to the actual resource use
- Tenant-aware version control for platform technology and application software
- Tenant self-service provisioning, administration and deprovisioning
- Tenant-aware error and disaster recovery

Multitenancy can be implemented at several levels of the software stack, from the allocation of virtual machines (VMs) to custom multitenancy in the application code itself. The higher the level at which multitenancy (sharing) is implemented, the finer the grain of increment of shared resources. At low levels, to implement elasticity, you have to add or remove the entire VM; at higher levels, you can add/remove just some threads or some local memory, or adjust up or down some priority. You are also able to track the resource needs at a finer grain level, so you are more responsive to the changing demands. Finally, with multitenancy implemented at the higher levels of the software stack, you can track the needs and provide needed resources differently to different parts of the application, which is impossible at the lower levels.

The degree and the scope of the sharing of resources among tenants vary from implementation to implementation. The more sharing, the more agile and inherently cost-effective the platform can be, but the more complex (and costly in terms of engineering and management) the platform implementation must be to preserve tenant integrity. The complexity of internal architecture for the fine-grain, elastic resource sharing among tenants also typically leads to nonstandard programming models. Until new standards for multitenant platforms emerge, this will continue to result in a lack of portability among the advanced multitenant platforms.

Sharing Versus Colocation

Sharing should not be confused with colocation. Consider the case in which multiple VMs are instantiated on one physical machine, but, once allocated, they remain permanently dedicated to their original tenants. In this case, the tenants may be colocated on the same hardware, but they are not sharing it, because, at no time, is any of the machine capacity reassigned from one tenant to another. This is a case of hosting an application over a cluster of VMs. (Traditional hosting is over physical machines; however, the use of VMs alone does not change the essentials of the pattern.). Colocation does not offer elasticity and, therefore, behaves like the traditional, inflexible and expensive dedicated hosting. When such isolated-tenancy hosting, paired with server virtualization, is offered one-to-many with some degree of self-service — the software is offered as a service, but — lacking the elastic sharing — not as a cloud service.

One-to-many offerings are available to all customers in the same manner (a "retail" model, as opposed to tailored offerings that are custom-made or adjusted for each customer). Traditional hosting is one-to-one, so each customer develops its own hosted environment. In the absence of self-service and one-to-many exposure to the market, the pattern is just traditional hosting.

With autoscaling, the VMs are allocated and released, as needed, in response to changing demand and based on predefined policies. The freed physical resources are reused for future VMs, then allocated for different tenants in different configurations. In this scenario, some computing resources (the hardware capacity) are shared between tenants, and this then is true elastic multitenancy (the shared-hardware type). Gartner's reference architecture identifies seven

types of multitenancy, including shared hardware. Typically, shared-hardware elasticity requires a technology layer that implements at least:

- Tracking of tenant use of resources
- Policies for resource allocation to tenants
- Policy-based allocation and release of VM images and storage to and from tenant compute clusters, in response to changes in utilization and demand

Examples of such technology include Microsoft Windows Azure Fabric Controller, IBM Workload Deployer, Red Hat OpenShift (Makara), RightScale and Amazon CloudMonitor with Amazon Auto Scaling. All of these platforms offer potential support of multitenancy through elastic horizontal scaling.

In addition, above the VMs or other *system* infrastructure, the *application* infrastructure that hosts the application (such as an application server cluster) must be horizontally scalable to automatically absorb injection of additional VMs and/or storage or automatically consolidate when resources are removed, without disturbing the running application. This is not a trivial capability, and many current application server offerings lack it. An application infrastructure platform that is not horizontally scalable cannot take advantage of the horizontally scalable system infrastructure. Hence, the application will remain without the benefit of elasticity or shared multitenancy, despite the potential for dynamic allocation of VMs.

The Two Main Types of Tenancy

Tenant-Enterprise

The concept of a tenant in cloud computing originates from the practices of the independent software vendors (ISVs) in SaaS — the early form of cloud computing. There, one application, developed is by an ISV and run by a SaaS provider (not always the same entity). The application is offered to many organizations, each of which is referred to as a tenant, which implies a degree of resource sharing with other tenants and isolated independence of each tenant from the others. For example, salesforce.com claims more than 90,000 customers (tenants) for its CRM application services. Logically, each tenant must operate in full isolation from others, but, physically, the tenant instances of the application share some resources (the degree and the depth of sharing differ with from implementation to implementation).

It is useful to look at this scenario as a collection of logical instances of the application running in a partially shared physical environment. Each instance serves a particular business organization (enterprise) that is a customer of the SaaS provider and a tenant of that application service. These are tenant-enterprises. Typically, a cloud platform that originates from a SaaS business implements multitenancy to support the tenant-enterprise use-pattern. Salesforce.com force.com is a classic example of this pattern. (In private cloud settings at large enterprises, multiple divisions of one organization may act as tenant enterprises in the organization.)

Tenant-Application

A cloud platform may host multiple application instances that are not all clones of the same application. Multiple applications — all used by potentially just one user organization or by individual consumers not affiliated with any organization — can also run in a partially shared physical environment and compete for resources in a manner similar to tenant-enterprise instances of a single multitenant application. In this scenario, application instances that share the underlying resources represent different business applications. (In the prior scenario, the

application instances were all renditions of the same application.) These are the tenant applications.

The underlying multitenant platform, aiming to provide a shared computing environment, must perform operations to deliver elasticity, resource use optimization, use tracking and other cloud characteristics to the competing tenant-applications. Service providers with the background of serving markets of individual consumers are typically focusing their design of multitenancy to support tenant-applications. Google.com is a typical example of such a shared multiple-application environment, although, recently, Google began offering Google Apps for Business (GAB), with some tenant-enterprise isolation.

Combining the Tenant Types in a Multitenant Environment

The separation between the tenant-enterprise and the tenant-application multitenancy is not absolute. In fact, most cloud platforms support a combination of both. A salesforce.com environment offers services of multiple applications, each supporting multiple customer organizations and all partially sharing one common computing environment. Here, one shared environment supports multiple tenant-applications and multiple tenant-enterprises. Google offers some of its applications (e.g., GAB) to business organizations and, thus, has tenant-enterprises and tenant-applications running in a partially shared environment.

However, the technology patterns for sharing computing resources among instances of the same application and instances that have little in common (because they're representations of different applications) are different. Most multitenant platforms are optimized for one or the other pattern. Google's recent challenges in delivering services to the U.S. government underscore these differences. Google was ultimately forced to build a dedicated community cloud environment to guarantee adequate security, reliability and other controls to U.S. government organizations.

Nested Multitenancy

One distinct characteristic of multitenancy that applies only to the tenant-enterprise use pattern is nested multitenancy. Any PaaS provider that supports multiple ISVs — which, in turn, support multiple customers — faces this challenge. First, the PaaS provider must provide the elastic resource sharing and tenant isolation to its tenants (the ISVs). Second, it must provide the shared elasticity, use tracking and isolation to the customers of the ISVs (the subtenants). The situation is further complicated by the nested requirements of version control.

The PaaS provider typically offers some nonintrusive version control for the PaaS services (the changes are backward-compatible and are rolled out without interrupting the operations of the running tenants). However, the ISV must deliver version upgrades and patches to its applications as well. In the absence of support for nested multitenancy from the PaaS provider, ISVs are forced to develop some multitenant version control as part of the application, acting in part as an application infrastructure provider (which is a challenge for most ISVs). Most ISVs should give preference to platform service providers that do support nested multitenancy.

Is an Individual User a Tenant?

The sharing of computing resources has been a feature of most application environments since the introduction of multitasking OSs and multiuser transaction-processing monitors in the 1960s. All modern OSs are multitasking, and the application servers are multiuser. Users in modern computing environments are all using a shared execution environment of the application server container and are also isolated enough to track their execution and security credentials separately and exclusively. However, we do not call individual users tenants.

Tenancy is associated with instances of applications, not user processes (transactions). Most of the resources that are allocated to users in a multiuser environment are allocated to short-lived individual transactions that a user initiates. As transactions are completed, all resources are released. There is only a minimal footprint of a user in an environment when the user is inactive. In contrast, a tenant (application instance) has a pool of resources that varies as demand changes, but it never goes to zero. A tenant has a permanent footprint in the environment, while the user does not (with the exception of some static state and metadata).

In multiuser execution, the sharing of resources among active user processes occurs inside the resource space of an application instance. In multitenancy, the underlying resources of the application container (shared container), operating system (shared OS) or hardware (shared hardware) are elastically shared among application instances. Application instances remain active for extended periods of time; user processes are short-lived. The resource requirements of the long-running application instances can vary, and will benefit from underlying elasticity. The resource requirements of a short-running user process are established at initiation and are fixed for the duration of the process; therefore, individual user processes would not benefit from the available elasticity of resources.

Even if a tenant-enterprise has only one user, it is still a tenant, because it is treated by the underlying multitenant infrastructure as a long-running application instance, with changing demands for execution resources. Technology designed to implement multitenancy manages application instances and leaves it to the application containers to support resource requirements of individual user processes.

Who Benefits From Multitenancy?

Users of services may not be aware of the difference between a cloud service (one that is internally resource-elastic) and a Web-hosted service (one that internally uses fixed, dedicated, unshared resources). The providers can see direct benefits of the elasticity of cloud computing, such as the improved total resource utilization and reduced costs of operations. For the users, the elastic cloud identity is less obvious: The service may not be less costly to the tenant, despite the cost advantages to the provider. An elastic cloud service can better cover spikes of demand, but many applications don't have such spikes. The tenant provisioning is faster in a shared environment, but each tenant gets provisioned only once, and is not exposed to this benefit in its regular operations.

For many users, the definitive characteristic of cloud service is its self-service feature and fast startup time, rather than its internal elasticity. This is where the users indirectly experience the benefits of cloud elasticity: the more elastic the platform is, the more agile and responsive the self-service is. Provisioning a new tenant or more capacity in a running shared environment amounts to a mere configuration change; while provisioning the same resources in a dedicated hosted environment requires obtaining and allocating a new batch of resources. The more resources are shared in a particular cloud, the faster it is to add a new tenant; therefore the self-service becomes more responsive.

Although internal elasticity may not be immediately evident to the users of software services, the elastic cloud services deliver a qualitative advantage of agility to both the users and providers of the services. When choosing a cloud service, users should make an extra effort to determine whether there is resource sharing, and to determine the depth of the shared resources. Greater internal "cloudiness" will deliver greater external agility over time. However, this should be balanced by other, more-immediate merit considerations. A platform with better performance, reliability, security, cost or customer support may be preferable, even if its internal elasticity is underpowered.

The Bottom Line

At the beginning of this research, we asked whether salesforce.com, Google.com and nationalgeographic.com are multitenant environments. Given that multitenancy is defined by partial sharing of physical computing resources among logically isolated instances of applications, we can offer the following answers:

- Clearly, salesforce.com CRM is multitenant, because multiple logical instances of the CRM application are running simultaneously in one shared physical computing environment (including the shared database and the shared execution container). Each instance draws computing resources from a shared pool, as needed, while retaining logical isolation. This is a tenant-enterprise multitenancy.

- Clearly, Google.com is also a multitenant environment, because multiple Google apps run in a shared computing environment, drawing resources from a shared pool of resources (including the shared use of BigTable), while retaining logical isolation. This is a tenant-application multitenancy.

- The question about nationalgeographic.com is rhetorical. The challenge here is not to understand the internal architecture of this particular website, but rather to understand how to determine whether any public website has multitenancy.

  Nationalgeographic.com and any other such websites may be multitenant if the software that implements its functionality is deployed on a multitenant platform. Some sites are developed using a multitenant PaaS; other sites might be offered by a provider that implements some level of custom multitenancy. These are multitenant. Other sites might be simply hosted in the data center of the provider or a third-party hosting provider. These are not multitenant. Often, the user won't know the difference between a site that is hosted without elasticity and a site that is deployed in a multitenant environment. However, the cost of running a multitenant site environment is lower, and the site is more likely to survive at least moderate spikes of demand. Although users may not be aware of the underlying multitenancy, providers and users enjoy a more effective environment when websites are deployed using underlying multitenancy. Over time, the multitenant sites will offer greater agility in facing changing demands and in managing their own changes within.

- Any Internet service that has internal multitenancy will offer more-agile scalability, faster repair and version control, faster startup time for new applications or new customer organizations, and relatively lower costs. However, a multitenant environment may offer less-assured privacy and performance characteristics. Users are advised to analyze their requirements and the multitenancy characteristics of the available services to compare the fit of the cloud service with the user's changing business requirements.

RECOMMENDED READING

Some documents may not be available as part of your current Gartner subscription.

"Gartner Reference Architecture for Multitenancy"

"Reference Architecture for Multitenancy: Enterprise Computing 'in the Cloud.'"

"Scalability, Elasticity and Multitenancy on the Road to Cloud Services"

REGIONAL HEADQUARTERS

Corporate Headquarters
56 Top Gallant Road
Stamford, CT 06902-7700
U.S.A.
+1 203 964 0096

European Headquarters
Tamesis
The Glanty
Egham
Surrey, TW20 9AW
UNITED KINGDOM
+44 1784 431611

Asia/Pacific Headquarters
Gartner Australasia Pty. Ltd.
Level 9, 141 Walker Street
North Sydney
New South Wales 2060
AUSTRALIA
+61 2 9459 4600

Japan Headquarters
Gartner Japan Ltd.
Aobadai Hills, 6F
7-7, Aobadai, 4-chome
Meguro-ku, Tokyo 153-0042
JAPAN
+81 3 3481 3670

Latin America Headquarters
Gartner do Brazil
Av. das Nações Unidas, 12551
9° andar—World Trade Center
04578-903—São Paulo SP
BRAZIL
+55 11 3443 1509

Gartner