Gartner

Research

Publication Date: 25 August 2010 **ID Number: G00205983**

Gartner Reference Architecture for Multitenancy

Yefim V. Natis, Eric Knipp

In the long term (beyond the next five years), computing "in the cloud" will require technologies and architectures that are natively designed for the cloud. In the short term (the next three to five years), intermediate models of technology and architecture will help the IT industry make the transition, although some of the intermediate technologies and architectures may not endure.

Key Findings

- Business applications offered as cloud services require multitenancy support in their enabling platform technologies; however, the established application platforms (application servers) do not possess this capability.
- There are seven main approaches to supporting multiple tenants in today's cloud environment, ranging from isolated tenancy (shared nothing) to custom application multitenancy (coded into the business application).
- The options having the most impact on multitenancy are the use of shared hardware (via autoscaled virtual machines) or the use of full-stack multitenant platforms (shared everything).
- Shared-hardware and shared-everything models of multitenancy each have their own differentiating strengths and sweet-spot use cases; both will improve over time, and will be offered and utilized in the long term.

Recommendations

- Conservative organizations that delay investment in cloud computing should ensure that their applications are designed to scale out. This is a good practice for most systems, and is a prerequisite to successfully migrating applications to cloud deployments. Proficiency with service-oriented architecture (SOA)-style software will also facilitate future integration of cloud application resources.
- Leading-edge enterprise IT organizations should begin adopting cloud application platforms now (for less-critical projects at first) to gain experience with the specifics of the public cloud application platform environment.
- Mainstream independent software vendors (ISVs) should no longer attempt to create their own cloud-enabling multitenancy support (custom multitenancy), but instead should plan to build their cloud-based application software using commercial cloud-enabled application platforms offered as a product or as a service.

TABLE OF CONTENTS

Analysis 3
1.0 What You Need to Know 3
2.0 Multitenancy 4
3.0 The Models of Multitenancy 4
3.1 Shared Nothing (Isolated Tenancy) 7
3.2 Multitenancy via Shared Hardware 7
3.3 Multitenancy via Shared OS 8
3.4 Multitenancy via Shared Database 9
3.5 Multitenancy via Shared Container 9
3.6 Multitenancy via Shared Everything 10
3.7 Custom Multitenancy 10
4.0 Strategic Outlook 11
5.0 Enterprise IT Impact 12
Recommended Reading 12

LIST OF FIGURES

Figure 1. Gartner Reference Architecture for Multitenancy 6

Gartner

1.0 What You Need to Know

Most software is designed to be sold to enterprises (user organizations) and deployed in the confines of that same organization, protected by internal security and integrity measures. The prevailing application platform programming models, such as Java Platform, Enterprise Edition (Java EE), .NET Framework and Spring Framework, have been designed to meet and excel with the requirements of this use model.

In-the-cloud applications are offered as a service to many user organizations (possibly tens of thousands or more). Such application services are often referred to as software as a service (SaaS). For providers to make this a business-viable model, some or most of the underlying computing resources must be shared among the many user organizations ("tenants"). Meanwhile, the user organizations must experience the service as if they were operating in a dedicated environment, without interference from other tenants. Each user organization must be able to customize the application to its own specifications. Yet, all customized instances of the application must still share common computing resources. This shared environment must support massive numbers of users (representing thousands or more tenants) and potentially unpredictable volatile demand. The current application platforms were not designed to support these nontrivial requirements.

To meet the new requirements of cloud computing, a new wave of platform design, platform programming models, platform products and aspiring platform technology vendors is emerging. At the same time, to avoid, or at least delay, discontinuous changes from single-tenant to multitenant platforms, innovative models of limited multitenancy are offered that preserve the traditional platforms and programming models. Two categories of offerings emerge:

- **Cloud-Enabled Application Platform (CEAP):** Cloud-enabled platform *products* are primarily application server functionality extended with support of multitenancy, horizontal scaling, metadata-based design tools, fine-grained use tracking and other specialized cloud capabilities.
- **Application Platform as a Service (APaaS):** Cloud platform *services* are not purchased as products, but rather subscribed to as services. The technology that enables the offering of such services is the same as the technology of a CEAP. In some cases, APaaS providers use proprietary enabling technology. In other cases, they use their own or a third-party CEAP, which is the enabling technology that is also available as a product to other buyers.

The new CEAP (product) and APaaS (service) offerings are the native cloud platform alternatives to the traditional enterprise application servers (see "Gartner Reference Architecture for Cloud-Enabled Application Platforms" and "Introducing SaaS-Enabled Application Platforms: Features, Roles and Futures"). An application project (at an ISV or an IT organization) may choose to:

- Buy a CEAP and build a cloud application in-house or hosted elsewhere.
- Subscribe to an APaaS and build a cloud application in the runtime environment of the APaaS provider.
- Build the entire cloud application stack custom in-house (not recommended for most projects).

Multitenancy is not only confined to the SaaS model, where an application provider serves multiple enterprises (as in the case of salesforce.com and other SaaS providers). There is also a scenario of many applications sharing a common computing platform (as in the case of Google and some other Web multiapplication providers). All the same principles of multitenancy apply in both cases, whether the resources are shared by tenant-enterprises, tenant-applications or both.

Users, IT organizations and ISVs must understand the risks and opportunities involved in the emerging long-term transformation toward cloud computing to avoid delivering underpowered or overprovisioned cloud solutions.

2.0 Multitenancy

Sharing common application computing resources among tenants is an essential characteristic of cloud computing, and is referred to as "multitenancy" (multiple tenants sharing common physical computing resources while remaining logically isolated). Depending on the use scenario, tenants may be user organizations or applications; in both scenarios, applications or instances of applications run in a physical computing environment that is shared between the multiple instances of some applications (each instance is a tenant image) and other applications, while presenting a logical computing environment that is dedicated to each tenant.

To fully deliver a multitenant service, the platform provider must implement the following characteristics (although not all are equally critical in all situations):

- Isolation of tenant data
- Isolation of tenant work space (memory)
- Isolation of tenant execution characteristics (performance, availability)
- Tenant-aware security, management, reporting
- Isolation of tenant customizations and extensions to business logic
- Tenant-aware application version control
- Tenant-aware error-tracking and recovery
- Fine-grained tracking and recording of resource use
- Ability to allocate resources to tenants dynamically, as needed
- Horizontal scalability to support real-time addition of new tenants or users
- Redundancy to support transparent hot migration in the event of resource failure

The more of these characteristics that are present, then the more agile, productive, seamless and cost-effective *may* be the tenant experience. However, the end effect will be a combination of some gains and some compromises, compared with the traditional custom IT process.

Gartner has described the multitenancy characteristics of SaaS (see "How to Evaluate SaaS Architecture Model Choices"). We now apply and extend that analysis to the context of cloud application platforms.

3.0 The Models of Multitenancy

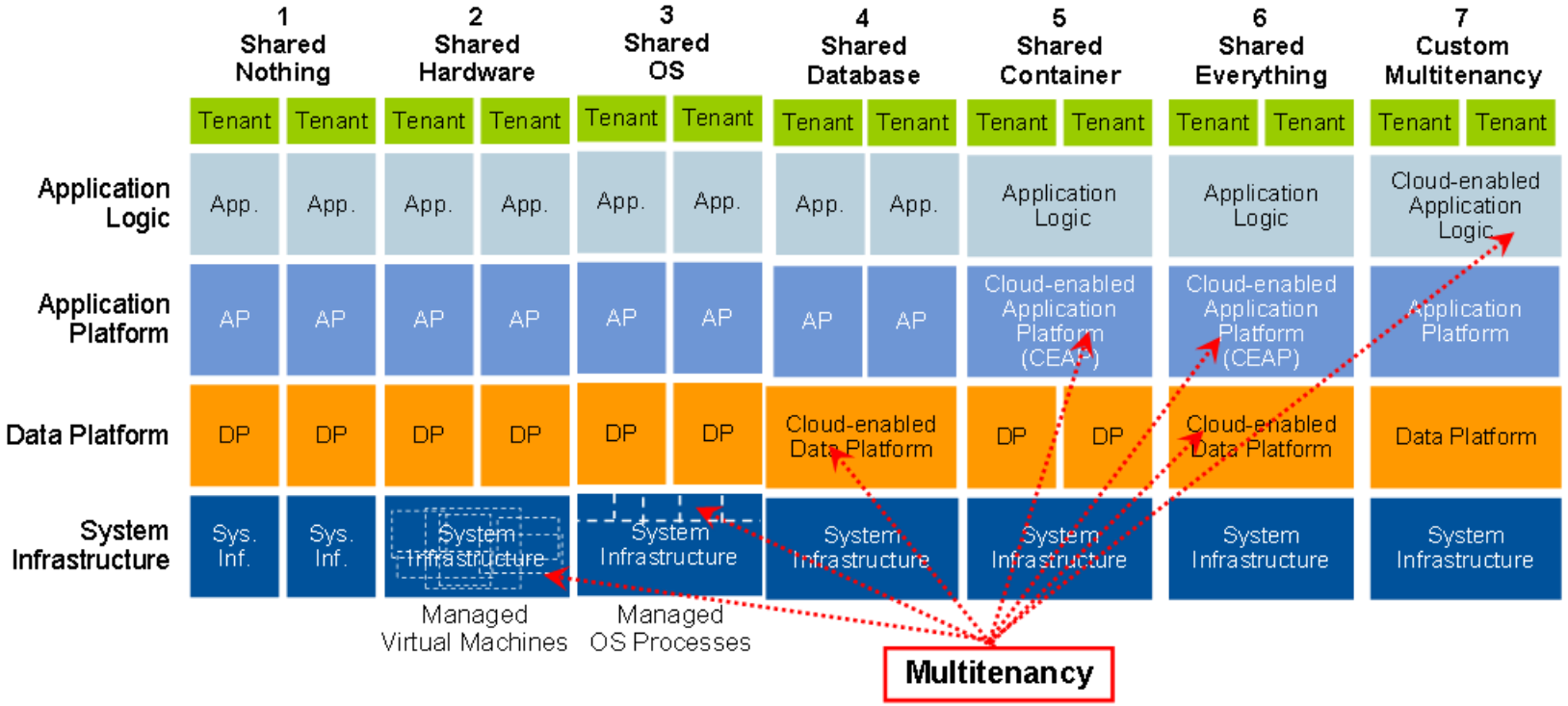
A traditional application deployed on-premises relies on an underlying application platform built around an application server, such as WebLogic, WebSphere, JBoss or .NET Framework. The

application platform, in turn, uses a data platform, typically built around a database management system (DBMS). The data and application platforms layer over the underlying system infrastructure is built around an OS, such as Linux or Windows. In the traditional on-premises environment, the entire stack serves one IT organization.

This core layered technology architecture, well-established in enterprise computing, applies to cloud computing as well. Cloud providers offer applications that are written for a particular underlying application platform, use a data platform and are hosted in a system infrastructure environment — just like on-premises business applications. Because the computing resources that underlie the business application are layered, the application's multitenancy can be implemented at any (or several) of the layers.

Figure 1 illustrates the models of multitenancy, with each based on a different approach to sharing the computing resources that underlie the application.

Figure 1. Gartner Reference Architecture for Multitenancy



Source: Gartner (August 2010)

Gartner

3.1 Shared Nothing (Isolated Tenancy)

Under this concept, each tenant has its own dedicated stack of underlying technology; nothing is shared. This is not true multitenancy, although users may experience it as multitenancy, because the same application is offered from the same external data center to multiple user organizations as a service. This model is similar in many respects to traditional hosting, because each user organization has its own dedicated stack of resources, and its own instance of the application that it customizes to the chosen degree. However, it is not the classic hosting model because, unlike hosting, the provider offers the same application (and, presumably, the same version) to all tenants, and the same application functionality is shared among all subscribers. Some shared self-service platforms for provisioning and managing the application may also be part of the offering. Hosting implies that each customer would have its own custom application version or a totally custom-developed application. Isolated tenancy is, in effect, hosting of clones of one application.

Isolated tenancy lacks agility or any potential for sharing or elasticity of computing resources. Adding a new tenant requires provisioning hardware, the OS and the rest of the software dedicated to the tenant, which can be a lengthy and expensive proposition. It is also not cost-competitive; there are no economies of scale, which would have come from sharing computing resources among tenants. (Isolated tenancy is a close cousin to plain hosting and delivers the benefits of hosting: the economies in real estate, staff, physical security, power, management and some hardware provisioning.)

Isolated tenancy is a highly limited substitute for cloud computing. However, in this model, applications and the underlying technologies that work on-premises also work off-premises, without a change. It may be a fast and entirely unintrusive initial step toward cloud computing for some ISVs while they are investing in cloud-specific technology architecture. The benefit is also the complete isolation of tenants all the way down through the hardware. Some larger enterprises have, thus far, been willing to forgo most of the early advantages of cloud computing in exchange for the fixed pricing and complete isolation of this model. Advances in the maturity and benefits of cloud-computing technologies and practices will likely cause most enterprises and ISVs to review this policy in favor of the more advanced forms of multitenancy.

Examples of this model include, but are not limited to, E2open Multi-Enterprise Business Process Platform, Microsoft Dedicated Hosted Exchange Server, Oracle CRM On Demand Private Edition and SAP CRM OnDemandand. (All these are applications offered as a service, but without multitenancy in terms of sharing any resources.)

3.2 Multitenancy via Shared Hardware

Under this pattern, each tenant has its own dedicated stack of technology, but the (virtual) machines are allocated from a pool of shared hardware using an autoscaling elasticity layer over some form of hypervisor-based virtualization. A tenant application instance operates on a number of virtual machines, and a control layer (such as Microsoft Fabric Controller or Amazon Auto Scaling Service) allocates additional virtual machines as needed. The virtual machines are released when no longer needed. The freed hardware resources are now available to be allocated with a virtual machine to another tenant. Thus, the hardware resources are shared between the tenants over time.

This model shares many characteristics of isolated tenancy, because most of the stack underlying the application is dedicated to a tenant; but it can be true basic multitenancy, as long as the underlying machine resources are shared in a dynamic manner among the tenants. This can be accomplished using some form of autoscaling and autoprovisioning, or (to a limited degree) using a variable-capacity virtual machine. *The presence of dynamic sharing is imperative*

and marks the difference between isolated tenancy (fixed resources, as in hosting) and multitenancy (shared pool of resources).

More than any other model, the shared-hardware multitenancy allows a low-cost immediate entry into cloud computing for established providers of applications, application platforms and data platforms. It can offer nearly full-backward compatibility, and, at the same time, a degree of savings and agility of a partly shared computing environment. However, the manner of elasticity is relatively crude in this model; the minimal increment of adding or subtracting computing power is typically a whole virtual machine. There is also a requirement to have a prebuilt, full-stack image for all applications to minimize the time it takes to provision, deploy and configure an additional functional machine into the running application environment. Finally, the new virtual machine and its middleware and application software must be connected to the running network and data sources. Accomplishing all this requires some time and, thus, the elasticity of the shared-hardware environment can be relatively slow.

The shared-hardware multitenancy model also assumes that applications are able to benefit from adding more virtual machines to their execution pools. This assumption is not always valid. Some applications are designed to run on a single machine, and have no provisions for being distributed; some distributed applications still can be spread only over a small number of machines before the DBMS becomes a bottleneck, reducing or eliminating any performance benefit of increasing the number of virtual machines. These applications are not able to benefit from a migration to cloud without a redesign.

Most enterprise deployment cases of shared-hardware multitenancy use the traditional application platforms supporting third-generation languages (3GLs) above the OS level. There is no additional investment for customization of applications by tenants, tenant-sensitive management or tracking. Many of the multitenancy characteristics are not implemented, or are implemented partially.

Advanced enterprise applications, which are stateless, distributable and with large computing resource requirements, are the most suitable for shared-hardware multitenancy. These applications are designed to scale out, and thus are able to benefit from the elasticity of this model; they use large amounts of computing power, making incremental increases in the size of a whole virtual machine appropriate. The established advanced enterprise applications are also expensive to rewrite, so the opportunity to retain the existing code and the underlying application infrastructure is a major advantage.

Examples of this approach include, but are not limited to, Adobe LiveCycle Managed Services, Engine Yard XCloud, Microsoft Windows Azure platform, salesforce.com/VMware VMforce and Tibco Silver.

3.3 Multitenancy via Shared OS

In this scenario, application instances of multiple tenants are allocated in one OS instance. Controlling software dynamically allocates additional application instance processes to tenants as needed, or releases them when no longer needed. The freed OS resources are available to be allocated to another tenant's new application process instance. Thus, the OS resources are shared by different tenants over time.

The shared-OS pattern has many common characteristics with shared hardware. In both scenarios, the application and its underlying application infrastructure are instantiated separately for each tenant and only (some of) the systems resources are shared. The applications are mostly backward-compatible, since the same precloud application infrastructure layer is used.

The multitenancy is managed below the application infrastructure level in the systems infrastructure. However, shared OS has some additional flexibility in allocating the system resources to tenants, compared with shared hardware, as it operates at a higher level. It costs less in time and resources to allocate an OS process than to allocate a virtual machine.

Examples of the shared-OS model include, but are not limited to, Google App Engine, Heroku, Joyent SmartMachine and Parallels Virtuozzo Containers.

3.4 Multitenancy via Shared Database

The shared-database multitenancy model implies that multiple tenants share one common logical DBMS, but each runs in a separate instance of an application server container. Sharing resources is limited to the DBMS, but implementing such an arrangement is relatively easy: the container or the cloud-enabled data platform keeps track of the tenants' identity and associates all data access with the primary key that identifies the tenant. The data isolation relies on privacy of the tenant key. Onboarding a new tenant only requires that a tenant key be generated and the DBMS access provided to the newly configured tenant container instance.

This approach provides limited elasticity and resource sharing, and is used mostly by the container vendors looking for a quick way to claim cloud-computing capability. However, it is a good first incremental step toward more-advanced forms of multitenancy. It can also be combined with the shared-hardware model on the container side, potentially further increasing the benefits of resource sharing without introducing a significant application design discontinuity.

Some examples of this approach include, but are not limited to, Engine Yard AppCloud and Google App Engine (where each application executes in an exclusively dedicated [unshared] Java Virtual Machine [JVM], while all applications share the same Google data platform).

3.5 Multitenancy via Shared Container

Under this concept, multiple tenants' transactions are executed in the same instance of an application container (specialized application server, a CEAP), but each tenant is associated with a separate logical or physical instance of the DBMS. Thus, the execution environment is shared among tenants, but the data platform is not. The premise of this approach is that DBMS isolation would ensure the integrity of tenants' data, while the shared application execution container offers advanced fine-grained elasticity and customization opportunities.

To ensure isolation of tenants inside a single instance of the application container (application server), the CEAP container must be designed to manage resource allocation to tenants (something no current application server is designed or called to do). The CEAP design must prevent accidental or fraudulent intrusions of the transactions of one tenant into the space of another tenant. This requires that the supported 3GL programming languages and frameworks be reduced in function, or that fourth-generation language (4GL) programming languages replace them to exercise greater control of the executing application. In many ways, JVM was a step in the same direction, eliminating the error-prone freedom in C++ to access all underlying resources. Most multitenant platforms in this category use a specialized virtual machine that interprets metadata to execute a transaction, just as JVM interprets byte code to do the same at a lower level. Using a metadata-based interpreter for the execution of applications enables excellent productivity and depth in customization of applications per tenant, while retaining the elasticity of a shared execution container. Such implementations, however, imply a completely new application development and execution environment — a potentially costly discontinuity for application developers and application ISVs. Some such implementations (including Force.com and LongJump), limited by the design of its metadata model, also may be limited in functional completeness and, therefore, suitable for some, but not all, software development projects.

Enterprise IT organizations' top objection to cloud computing is that by allowing outside data centers to handle their business data they are potentially exposing their data to competitors or other intruders. The shared-container multitenancy approach can preserve the physical DBMS isolation, and partially mitigates this concern. Theoretically, the physically isolated DBMS of a tenant could be instantiated on-premises (thus further mitigating the data integrity objection). However, the performance overhead of the interaction between an in-cloud application container and on-premises application DBMS would make such an arrangement impractical for most users. A multitenant, metadata-driven application container can be a flexible, responsive and cost-effective cloud platform for new business applications, but at the price of discontinuity in the programming model.

Examples of this approach include, but are not limited to, Cordys' Business Operations Platform, Magic Software Enterprises' uniPaaS and Relationals' LongJump.

3.6 Multitenancy via Shared Everything

This model is an enhanced variant of the CEAP-based, shared-container multitenancy model. The CEAP model is still the center of action; however, in the shared-everything multitenancy model, multiple tenants share the application execution platform and the data platform under the management control of CEAP. This model shares the benefits and costs of the shared-container multitenancy model, and adds precision in managing shared resources at the cost of eliminating the possibility of physically isolated management of tenants' data. Only a single logical instance of the data platform is allocated for all tenants; tenant data isolation is managed in the CEAP and possibly in the cloud-enabled data platform as well. In addition to the finer granularity in resource sharing, this model offers a rapid and low-cost, on-ramping and off-ramping of tenants, applications and users — an important characteristic for an environment that potentially manages many thousands of tenants.

Examples of this model include, but are not limited to, Apprenda, ForeSoft dbFLEX, Intuit Partner Platform, Rackspace Cloud Sites, Rollbase Platform, salesforce.com Force.com, WaveMaker and Zoho Creator.

3.7 Custom Multitenancy

Some SaaS-style applications have been designed using conventional programming languages over conventional third-party platforms, yet are offered as a multitenant service. Typically, such applications implement the principles of multitenant isolation as part of the application code. A skilled development team can implement isolation and customization algorithms in the business application that may compete with the general-purpose multitenant platforms. However, the costs of the broad skills that are required, as well as the burden of the long-term responsibility for this system software being embedded into a business application, are high. Early SaaS providers had no available third-party CEAP options, and were forced to create the custom multitenant applications. Some of today's multitenant platforms (Force.com, LongJump and DBFlex) are offered by vendors that first developed the platforms as part of their applications, then extracted and productized the platform code from the business application code. Today, with a growing number of options for CEAP and APaaS, most organizations should avoid this model of multitenancy.

Examples of this model include, but are not limited to, Google Apps Premier Edition, Microsoft Dynamics, Microsoft SharePoint Online and Oracle CRM On Demand. (All these are multitenant cloud applications that have internal custom support of multitenancy that is not available as a platform for independent use.)

4.0 Strategic Outlook

Of the six approaches to multitenancy described in this research, two stand out as having the most impact:

- **Shared-hardware multitenancy** (typically using virtual machines) enables a low-cost, low-impact transition to cloud computing for mainstream enterprises in some scenarios, because it preserves the programming model and platform technology that are well-established in current enterprise IT organizations (although limitations apply, excluding most of the simple monolithic applications). Not surprisingly, Microsoft, Oracle and IBM, burdened with their responsibilities for their installed bases, and with the investments they have in their current platform technologies, chose this option for their entrance into enterprise cloud computing. This model is particularly suitable for projects where the current on-premises workload is moved to the cloud; however, many such approaches will result in simple hosting, because the autoscaling of a virtual machine environment is not applied or because the migrated applications are not designed to automatically adopt to the dynamically changing underlying network of virtual machines. *This model is the preferred **c**hoice for migrating current workloads to cloud platforms.*
- **Shared-everything (or shared container, which is similar) multitenancy** enables the application and the platform technology to be designed and implemented with the full recognition of the specifics of cloud-computing context, demands of multitenancy and horizontal scaling, and with cloud-user requirements. In the long term, computing in the cloud will require technology that was designed for the cloud and will offer effective multitenancy, horizontal scaling, use tracking, massive scalability, accessible productivity tools and self-service management — natively. The shared-everything model is the implementation of this principle. It is not surprising that cloud innovator vendors, such as salesforce.com, Heroku, LongJump and many others, took this approach. Native to the cloud, this model offers greater flexibility in provisioning, customization and optimization of applications, but at the price of a discontinuity in application programming models, platforms and skills. *This model is the preferred choice for new workloads designed from the ground up for cloud context.*

The choice on the way to the cloud between incremental adjustments and radical rearchitecture is also a choice between well-established enterprise software vendors versus the new enterprise providers. For the next three to five years, the industry is likely to witness the "war of multitenancy models." It is likely that the shared-hardware offerings, backed now by technology powerhouses, will be made increasingly efficient, agile and user-friendly. Meanwhile, the new cloud platform providers (the enterprise market underdogs) will likely attempt to establish some new standards for cloud programming to reduce the risks of lock-in with their offerings. Both types of offerings will improve, and their functionalities will be partly similar. However, in the end, we expect Microsoft, IBM, Oracle and other enterprise platform technology vendors to endorse the shared-everything, CEAP-based model, while also retaining the shared-hardware, traditional application server-based model. This will support the gradual transition for established customers and full cloud power for new users, and will help establish a broadly present hybrid cloud environment

It is notable, also, that most providers combine multiple models in their cloud environments; some full-stack multitenant platforms also use virtual-machine models to manage low-cost, upward scalability for the total environment. The largest customers of SaaS offerings sometimes request isolated tenancy, forgoing the elasticity in exchange for stable costs and completeness of isolation. *The leading cloud/SaaS platform providers will creatively combine all the available models of multitenancy, because each has its own contributing advantage for some use scenarios.*

To achieve this outcome, certain established vendors will acquire innovators, and some innovators will merge to create an alternative cloud environment. The result of this market consolidation will likely make current application infrastructure vendors stronger, and will propel a new vendor or two into a position of leadership in the enterprise application infrastructure markets.

5.0 Enterprise IT Impact

Enterprise IT organizations are facing an impactful change in their computing landscapes:

- New ISV solutions will emerge that are only available in the cloud.
- New multitenant APaaS will offer higher levels of productivity, and a new business model of software engineering with the new cost structures:
    - Elastic allocation of resources will reduce the costs and risks of spikes in business activity and will become a requirement for most IT organizations exposed to the volatile (Web or other) business demands.

Gartner believes that, despite the perceived risks, mainstream enterprises gradually will adopt cloud computing. In the process, the enterprise IT department will be exposed to a changing platform environment that will require new skills, as well as a new business model for the business use of IT.

In choosing the multitenant cloud platforms, there is no one right answer. Enterprises should be prepared to adopt cloud models incrementally. They should look for safety provided by the traditional programming models for some projects, yet also embrace the transformational opportunities of the new, native cloud-computing models.

Contributing to this research were Tom Bittman, Michele Cantara, Robert DeSisto, Nickolas Gall, Benoit Lheureux, Ben Pring and Roy Schulte.

RECOMMENDED READING

"Gartner Reference Architecture for Cloud-Enabled Application Platforms"

"Application Infrastructure for Cloud Computing: A Growing Market, 2010"

"VMware and Salesforce.com: The Beginning of a Beautiful Friendship?"

"Who's Who in Application Platforms for Cloud Computing: The Cloud Specialists"

"Who's Who in Application Platforms for Cloud Computing: The Enterprise Generalists"

"Cool Vendors in Application Platforms as a Service, 2010"

"Microsoft AppFabric: A Platform for the Cloud Era Is Under Construction"

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

"Predicts 2010: Application Platforms for Cloud Computing"

"Predicts 2010: Application Infrastructure for Cloud Computing"

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