WHITE PAPER

Meeting the Needs of Database Management for SaaS: Oracle Database 12c

Sponsored by: Oracle Corp.

Carl W. Olofson
September 2014

IDC OPINION

The move of ISV applications to the cloud is growing from a trickle to a stream to a torrent. The pressure is on for ISVs to become software-as-a-service (SaaS) providers. Most existing ISVs, however, have complex applications that operate using on-premise dedicated server deployment and on dedicated database configurations. There are multiple solutions available for moving the application code to the cloud, but not so for the database. Fortunately for Oracle Database application ISVs, there may be a simple way forward.

The situation may be summarized as follows:

- The cloud requires the capability to deliver elastic scalability, multitenancy, and configuration agility in order for a SaaS offering to be successful.
- A variety of cloud service providers can help deploy application code so that it can meet these requirements through a combination of operating system (OS) and application support.
- Databases have special constraints that make addressing cloud requirements difficult. New technologies are emerging in response to this need, but for an existing ISV to move to the cloud and adopt one of these emerging database technologies would involve substantial code changes.
- Oracle Database 12c with the Oracle Multitenant option is designed to address the requirements of database deployment in the cloud, so for ISVs that base their code on Oracle, transition to a SaaS model should be fairly smooth.

IN THIS WHITE PAPER

This white paper discusses the challenges involved in deploying a database in the cloud and the technology required to do so successfully. It also speaks to the challenges faced by ISVs that wish to deploy their database applications in the cloud and realize the economies of scale that such a deployment model promises without severe challenges in terms of management and administration. It reviews the requirements for a database platform that can serve the needs of such ISVs and how Oracle Database 12c with the Oracle Multitenant option meets those requirements.

SITUATION OVERVIEW

As software-as-a-service options become more robust, enterprises are choosing to reduce the costs and risks associated with on-premise applications. As a result, the move is on to shift application functionality from on-premise deployment to the cloud. New application vendors that offer business functionality in the cloud that is cheaper and easier to manage and that reduces their clients' capital expenses are emerging.

Yet many of the SaaS offerings lack the functionality of on-premise applications that have evolved over decades. Such applications, designed for on-premise operations, have many nuanced characteristics developed over the years in response to customers' requirements and a deep understanding of the business problems they confront. The problem is that they are not designed with the cloud in mind, and so application vendors are confronted with either finding a well-designed cloud application environment that is compatible with their applications or fundamentally rearchitecting the application. The latter is clearly the less preferable course, at least in the short term.

Requirements for Cloud SaaS

So how different is the cloud from on-premise deployment, and what are the requirements for cloud-based SaaS? To operate in a way that takes advantage of economies of scale while still delivering acceptable levels of performance and reliability, SaaS offerings run in environments specifically designed for the cloud, featuring the following characteristics:

- Elastic scalability
- Multitenancy
- Configuration agility

These three characteristics combine to ensure that SaaS is always delivered in the most efficient and cost-effective manner possible. Also, for a SaaS provider to offer consistent performance in conformance with one or more available service-level agreements (SLAs), it must ensure both access-level and data-level security, and the application must always be available.

Elastic Scalability

Typical on-premise deployments involve a dedicated server, or set of servers, for each application, with dedicated database servers and storage as well. Because these resources are fixed, they must be allocated to serve the "high-water mark" for each application or database, and they cannot be shared, pooled, or reallocated. As a result, the classic on-premise deployment approach results in very low levels of average utilization and therefore a very low work-to-cost ratio. This is a key reason for enterprises to adopt cloud computing, both on-premise and in the public cloud.

A key characteristic of cloud for boosting utilization rates is that of elastic scalability. Elastic scalability is the ability to scale a given workload up or down by adding or removing resources on a dynamic basis. Enabling this capability is the cloud provider's responsibility, but in some cases, the application may need to be modified to adapt to the presence or absence of processors, memory, and so forth if the application is designed for grid-style deployment. Most existing applications that move to the cloud can be accommodated in this manner by operating them within a hypervisor environment, which virtualizes the resources and makes their allocation transparent to the application. If the application already runs in an application server environment, such as a Java container system, this capability can be realized in an even more efficient manner.

Multitenancy

Unlike an on-premise application deployment, which serves a single account through a dedicated system of servers and storage, a SaaS provider serves multiple accounts using a common pool of servers and storage. In doing so, it is necessary that each account, or tenant, seem to be alone in the system and that the other accounts, or tenants, not be visible and, from the perspective of the user, not even exist. This means not only that other tenants are undetectable but also that anything they do to impact resource availability, such as surges of CPU-intensive operations that could diminish processor performance or surges in I/O that could diminish storage performance, must be buffered by the environment so that these sorts of effects, sometimes called "noisy neighbor" effects, do not take place. Multitenancy is key to achieving the economies of scale that ensure consistent profitability for both the cloud platform provider and the SaaS provider.

Configuration Agility

Elastic scalability and multitenancy are achieved, in part, through configuration agility; that is, by the ability to dynamically allocate and deallocate resources (especially servers and storage) to and from workloads and to move workloads around from server to server as needed to ensure even performance and eliminate the "noisy neighbor" effect. This is accomplished at the application level through virtualization at two levels. One such level involves subdividing servers and storage into virtual servers and volumes, and the other involves insulating application instances from servers and storage so that they may be moved around and reallocated without disturbing application operation.

These capabilities are achieved at the application level by technologies that include hypervisors (which run operating system instances inside virtual machines) and object storage (which virtualizes files and volumes, enabling them to be managed across a storage array). From the application instance perspective, the effect should be that the application is alone on the servers and volumes that it uses, even though those virtualized servers and volumes are elements of physical servers and volumes, their operations may be spread across many volumes that seem like one, and many application instances may run in one server that operates as if it were many. Ideally, this should work without requiring any changes to application code designed to run on dedicated servers and storage.

Discrete SLA Management

Most SaaS offerings include multiple subscription levels, depending on the level of performance required. To make this work, the SaaS provider needs to be able to set parameters on a tenant-by-tenant basis that establish limits on resource allocation and, in some cases, process scheduling. Associated resources, including databases, need to offer similar tenant-level parameters when shared resources are used.

Database Cloud Challenges

For applications, the approaches described previously enable the key cloud characteristics that are required for a successful SaaS. If the applications run on a database, however, this is not enough. It must be possible to deploy the database in a way that also supports those characteristics. The problem is that, for a variety of reasons, a database management system (DBMS) cannot be deployed in a manner that supports elastic scalability, multitenancy, and configuration agility without explicit support in the DBMS itself.

Most DBMSs Expect Fixed Configurations

Most DBMSs are designed to run on a fixed server or cluster of servers with a fixed storage resource. They are optimized at a low level to ensure high levels of performance by the way they manage data in memory and on disk. As a result, they cannot be dynamically moved about, their resources cannot be virtualized externally, and they cannot modulate their performance to suit the SLAs of different tenants without support in the database server software itself. This inability to move about from server to server, and to dynamically allocate and deallocate resources, impedes elastic scalability.

OS-Level Virtualization

Using a hypervisor as the means of virtualizing database resources is problematic because, by its nature, the hypervisor's virtualized environment looks real to the software running under it. Overall application performance is often dependent on database operational performance. Database operations and optimizations are based on an assumption that the DBMS has sole control of the underlying infrastructure, including a dedicated database server configuration. Because of this assumption of sole control, the DBMS can encounter unexpected delays and resource conflicts, especially if other database instances are running in other VMs on the same server. A better approach is to have DBMS software controlling the server and acting as the host for client databases running on it. This approach can support dynamic resource allocation, movement of client databases from one server to another, and performance isolation.

Application-Level Multitenancy

Some SaaS and PaaS environments offer multitenancy within shared database instances at the application level. They do this by using features of the DBMS that allow discrete separation of the data so that each tenant (defined as a group of users) sees, and knows about, only its data. The problem is that if per-tenant SLAs are desired, and they usually are, there is no straightforward way to enforce those SLAs for each tenant at the database server level, unless the tenants are physically separated — that is, running on separate servers. Also, it is not possible for tenants to change their schemas separately; all the tenants operate on a single set of table, column, foreign key, and index definitions. This is because the DBMS does not know there are multiple tenants. The only mechanism for varying performance for a given tenant (which the DBMS sees as groups of users) is at some cumbersome level such as user or group priority. Running the tenants on discrete database servers, however, denies a critical level of economy of scale that the cloud promises.

Noisy Neighbors

If multitenancy is accomplished at the application level and multiple tenants are assigned the same database server, tenants can experience performance problems due to noisy neighbors because the only means of managing multitenancy is priority setting, which does not ensure a given SLA, only that higher-priority users will get more of whatever resources are available. If one tenant is running a long, resource-hungry query, the other tenants still suffer.

Ease of Adoption

There are numerous approaches to solving the cloud database problem, including sharded open source RDBMS deployments on clusters, use of DBMSs with unusual data structure and access requirements, and NoSQL DBMSs. All of these approaches would require substantial code changes for existing applications to use them. If, on the other hand, we have a DBMS that internally supports multitenancy, elastic scalability, and configuration agility, then complex application code changes are not necessary. Ideally, the DBMS should be 100% compatible with the application SQL as well.

Requirements of a Cloud Database

In summary, a database that is expected to support a SaaS offering in the public cloud must have the following characteristics:

- **Explicit internal support for elastic scalability.** In the database case, this means enabling the addition of resources to or removal of resources from a database instance without requiring changes to the database or its metadata or the application. It also includes the ability to "spin up" a new database instance on demand. Having a cluster-based architecture can enable a DBMS to satisfy a part of this requirement, but clustering alone is not enough.
- **Explicit internal support for multitenancy.** This includes support for an explicitly defined level of consolidation density (the number of tenants supported on a given server) as well as various aspects of isolation including performance, resource consumption, and data. This requirement also includes ensuring that each tenant can be assigned a discrete SLA and that the SLA can be met regardless of "noisy neighbors."
- **Explicit internal support for configuration agility.** In the database case, this means enabling a database to move from server to server without changing either the server or the application.

All these requirements are to be met by the DBMS as features of the database server software.

ORACLE DATABASE 12C WITH THE ORACLE MULTITENANT OPTION

Oracle has developed an option for Oracle Database 12c that addresses the requirements of a cloud database. It is called Oracle Multitenant, and it enables separation of the database metadata into two parts: the server operational part and the schema (or application)-specific part. The former is the basis for what is called the "container database." The latter defines a "pluggable database." Many pluggable databases (up to 252 in the current release) can reside on a single container database. Pluggable databases can be moved easily from one container database to another without disrupting operations. Pluggable databases can be spun up quickly as exact copies of other pluggable databases. Pluggable databases that are on the same container database can share resources; the container governs such sharing.

Oracle Database 12c with Oracle Multitenant addresses the requirements of a cloud database in the following ways:

- **Elastic scalability.** Resources such as memory and processors are allocated at the container database level to pluggable databases and can be added or removed at will. When storage is added to a container database, it becomes available to all the pluggable databases within the container. If a database requires more resources than a given container can assign, the database can be moved transparently to another container. This capability is enriched when the Multitenant Option is used in conjunction with the Real Application Clusters (RAC) option, enabling the addition of nodes to a database configuration as well as the movement of pluggable databases around the cluster.
- **Multitenancy.** Each pluggable database operates as if it were a standalone database. Users of one pluggable database cannot see another. Also, users of a pluggable database can change its schema without disturbing other databases of the same general type, yet they can all share system resources. Settings at the container level can ensure satisfaction of the SLA for each pluggable database, and the container can adjust performance to insulate one pluggable database from overactivity by a noisy neighbor pluggable database on the same container.
- **Configuration agility.** Because physical resources are defined at the container level, they can be moved around and made available to pluggable databases in a transparent way. Also, the pluggable databases can be moved from smaller servers to larger servers when more power is needed, and back to smaller servers, with no impact on the application.

In addition, it should be noted that this is otherwise a completely normal Oracle Database, so ISV application code designed to work with Oracle Databases should work without any modification. As a consequence, Oracle application ISVs should face very minimal work in moving to the cloud, especially as far as the database is concerned, as the Oracle Multitenant option is designed to deliver, from the data management perspective, something close to an instant SaaS architecture.

An Optimized SaaS Platform

For a variety of reasons, Oracle ISVs may find that IaaS environments, which feature generic servers and storage, may lack the specific scalability and tenability necessary to deliver SaaS based on Oracle Database in the most manageable and cost-effective way. For this reason, a better choice may be a custom deployment, perhaps on a hosting site, to use just the right hardware to get the most out of Oracle Database. In making such a choice, it makes sense to strongly consider Engineered Systems from Oracle, including Exadata or Oracle SuperCluster, which are engineered specifically to maximize Oracle Database performance and manageability. While any standard computing platform is suitable for Oracle Database12c with Oracle Multitenant, the Oracle Engineered Systems may be best for overall performance and ease of administration.

FUTURE OUTLOOK

As database technology becomes increasingly sophisticated in its use of cluster-based in-memory database operations, such technology will become key to the best performance and simplest database management in the cloud. Oracle already offers in-memory database capability and is likely to continue to develop this technology. At the same time, more and more DBMS vendors will offer multitenancy support for the cloud, but one thing that will not change is the fact that for ISVs already running on Oracle, among all available options, a move to Oracle Multitenant requires the least amount of change to the application.

CHALLENGES/OPPORTUNITIES

As database technology evolves, greater and greater capabilities will emerge to exploit the possibilities of database management in the cloud from both established and newly emerging vendors. It is incumbent upon Oracle to continue to keep ahead of the curve, especially in support of Oracle's existing enterprise application ISV partners.

CONCLUSION

Increasingly, enterprises are looking for ways to reduce their ongoing operational costs and capital expenses in relation to IT. One key effort involves moving application functionality from existing on-premise deployments in the corporate datacenter to the cloud via SaaS application providers. Enterprise application ISVs are under a great deal of pressure to keep ahead of their customers by becoming SaaS application providers themselves.

But moving to the cloud poses challenges. In terms of application execution, a wide range of services are available from a variety of vendors to enable all but the most arcane applications to make the leap from on-premise to the cloud, except for the problem of the database. Challenges posed by the need for elastic scalability, multitenancy, and configuration agility in the database could force such application ISVs to adopt new database technologies that require massive coding changes. There is an alternative, however, for application ISVs that have based their applications on the Oracle Database. Oracle Database 12c with Oracle Multitenant is designed to address the requirements of the cloud while requiring little if any change to application code for existing applications.

Oracle application ISVs considering a move to the cloud should take into account the following:

- Review all the available hosting services to find the one that makes the most sense in terms of operational support, performance, flexibility, and fees that can fit in the framework of your planned subscription price.
- Consider the extent to which your customers vary in terms of resource requirements now and in their future plans and make your own plans for addressing those needs accordingly, including both resource levels and the need for flexible management of those resources.
- Consider Oracle Database 12c with Oracle Multitenant as the means of migrating your application with all its database functionality intact, meeting the requirements of a cloud application, but with no or nearly no code changes required for database support.
- Press Oracle to continue to innovate and strive for leadership in the increasingly competitive area of cloud database deployment, since its competitiveness will certainly impact your competitiveness.
- Maximize economies of scale in terms of both system efficiency and operational simplicity by consolidating SaaS deployments on Oracle's Engineered Systems, Exadata Database Machine, or Oracle SuperCluster.

About IDC

International Data Corporation (IDC) is the premier global provider of market intelligence, advisory services, and events for the information technology, telecommunications and consumer technology markets. IDC helps IT professionals, business executives, and the investment community make fact-based decisions on technology purchases and business strategy. More than 1,100 IDC analysts provide global, regional, and local expertise on technology and industry opportunities and trends in over 110 countries worldwide. For 50 years, IDC has provided strategic insights to help our clients achieve their key business objectives. IDC is a subsidiary of IDG, the world's leading technology media, research, and events company.

Global Headquarters

5 Speen Street
Framingham, MA 01701
USA
508.872.8200
Twitter: @IDC
idc-insights-community.com
www.idc.com