Database Virtualization

The Next Wave of Virtualization

Last Updated: December 12, 2014

Introduction

Virtualization has taken the computing world by storm. Server virtualization was the front of the virtualization wave, but we are now seeing a strong push to virtualize storage and networks. Many pundits now believe that database virtualization is the next big technology wave to hit the computing landscape.

While SQL databases are the most popular tool for online transaction processing (OLTP) workloads, they are also the most challenging databases to virtualize because of the way they tightly link the processing and data on a single physical server.

This document looks at different approaches to database virtualization and the benefits each approach derives. It also looks to the future of database virtualization and which database architectures are best suited to virtualization.

What is Virtualization

Virtualization is the creation of a virtual version of something (operating system, storage devices, network, etc.) that can be deployed and managed in a more fine-grained manner than the physical item itself. For example, a single physical server (or machine) can be sliced into various virtual servers (virtual machines or VMs), each embodying various resources (memory, disk, CPU cores, etc.).

Instead of dedicating a server to a specific function, which may not fully utilize the capabilities of that server, it can be sliced into various virtual machines. The full capabilities of that physical server are then *allocated* to the virtual machines (VM) as needed. For example one VM might have a large slice of the available memory, while a larger amount of disk space might be allocated to another VM. If one of the virtual machines is running low on a specific resource—e.g.memory—more memory can be allocated to that VM on the fly. Unlike a physical server, IT can allocate resources to virtual machines dynamically, enabling a greater degree of flexibility and more granular management.

Cloud Computing & Virtualization

Cloud computing and virtualization are synonymous. Cloud computing is based upon virtualizing and allocating compute, storage and network services in a shared multi-tenant environment. Virtualization is a key enabler for cloud computing. At the same time, cloud computing is also a powerful force pulling virtualization into the enterprise. The two are intimately linked, and enjoy a symbiotic relationship.

The Benefits of Virtualization

"Our focus has been that many databases are accessible and manageable as if they were a single database. Virtualization provides a common framework for better availability, scalability, manageability and security."

– **Noel Yuhanna, Principal Analyst, Forrester Research**

Despite some minor nuances, the benefits of virtualization are common regardless of what is being virtualized: server, storage, network or database. The following is a list of the benefits delivered by virtualization:

1. Flexible Deployment: By dealing with compute resources on a logical level, instead of a physical level (individual servers), you can run smaller applications in a fraction of a server, while larger processes can be run across multiple servers.

2. Rapid Deployment: By creating a virtual image of a server, you can rapidly deploy that image on any server, avoiding the nuances of the various physical servers. Instead of running an installation process on each server, you just install the VM on that server once and then copy any number of application images on top of the VM.

3. Server Consolidation: By allocating compute resources as needed, you avoid having dedicated, but underutilized, servers. Those server capabilities can be safely used across multiple independent applications, or applications can be run across multiple physical servers.

4. Business Flexibility: In the old days, each new development effort would entail purchasing new servers to run it, and they needed to accommodate both growth and peak load scaling of that new application. Now you can simply run that application on unused compute resources, and if and when you need to scale it, you simply allocate those resources from an unused pool of virtualized resources.

5. Energy/Cost Savings: By consolidating your applications on fewer physical "shared" servers you can unplug unutilized servers, reducing your energy use and your energy and server costs.

6. High-Availability: By enabling multiple virtual instances of an application to run on separate servers, loss of physical servers simply means that the load is handled by the remaining instances, until new instances can be launched.

7. Management Automation: By managing your application at a logical level (versus the physical servers) and abstracting away any specific server differences, IT management, including processes such as backups, are dramatically simplified and can then be automated.

8. Improved Quality of Service: Cloud environments result in a higher degree of application density. This can result in the noisy neighbor problem, where a neighboring application is consuming such a high degree of computing or network resources, that it reduces the performance, or quality of service, of the nearby applications. Mobility, the ability to move an application without bringing it down, enables cloud management technicians to move applications away from noisy neighbors, thereby ensuring a high quality of service and customer satisfaction.

These are some of the key benefits recognized through virtualization. This is complicated by the fact that there are degrees of virtualization.

What is Database Virtualization?

Database virtualization means different things to different people; from simply running the database executable in a virtual machine, or using virtualized storage, to a fully virtualized elastic database cluster composed of modular compute and storage components that are assembled on the

fly to accommodate your database needs. We will look at each type of virtualization and the benefits they provide.

Virtualization is anathema to traditional SQL-based OLTP databases. These databases tightly integrate the compute, caching and storage in order to: (a) optimize performance; (b) coordinate locking to ensure that the database remains consistent; (c) provide "copies" for fail-over or high-availability. Building a fully virtualizable database management system (DBMS) is a huge undertaking. However, deploying and using a fully virtualized DBMS is actually quite easy, since it eliminates a collection of deployment challenges required to scale-put single instance DBMS.

Database Virtualization Challenges

Traditional databases operate on a single physical computer. They tightly integrate the compute, caching and storage functions, operating as a single unit on a single server in order to optimize performance. This runs completely contrary to virtualization where the idea is to separate the logical (database functions) from the physical (the server). So the biggest challenge in virtualizing databases is to abstract logical processes away from the physical hardware, while still maintaining competitive performance.

If we look at the underlying database architectures, we find that the shared-nothing architecture is the antithesis of virtualization. Its name says it all: "share nothing". This architecture creates database units that are isolated, tightly tied to physical servers and that do not "share". Yet virtualization, at its core is the sharing of compute processes across a pool of compute resources. Clearly, retrofitting the shared-nothing database architecture, in order to virtualize it, is a non-starter, unless you are willing to settle for a grossly sub-optimal degree of virtualization.

Shared-data databases—also known as shared-data or shared-everything databases—are far more conducive to virtualization. They start with the premise that the data will be shared across an arbitrary collection of database servers. The underlying architecture must include a distributed lock manager to coordinate the locking processes of these various servers, ensuring that they do not collide with each other, which would result in inconsistency. This distributed locking is integral to shared-data databases.

The traditional name of these databases was shared-data, because they enabled multiple database nodes to share a single data repository (disk). However, this creates an I/O bottleneck that is simply unacceptable. Modern shared-data databases, like Oracle RAC and ScaleDB, employ a different model. They distribute the data across an arbitrary number of mirrored servers. This approach offers a number of advantages, including: (1) creating a pool of RAM cache to supplement the database nodes; (2) spreading the disk I/O across an arbitrary number of servers enables reads and writes to be fanned out to multiple disks, overcoming the single disk head bottleneck; (3) queries can be parallelized across multiple storage servers, with the database node then aggregating the results from the storage nodes. This process analogous to map-reduce, but it is performed within the ACID (atomicity, consistency, isolation and durability) model.

The shared-data database architecture is ideally suited for virtualization. It inherently provides an abstraction layer between the data and the compute, enabling them each to operate on an arbitrary number of servers or virtual machines. The distributed lock manager also coordinates the interaction between the instances of the database, ensuring that the database remains consistent. In short, once you have implemented a shared-data database architecture, you have solved all of the virtualization challenges; you have created a virtualized database.

Since the cloud, both public and private, is driving the virtualization trend, the next question is whether the shared-data database has been optimized for cloud infrastructures. Those cloud infrastructures are almost exclusively based on Ethernet. Oracle RAC requires Infiniband for a variety of reasons including its RDMA capabilities. This means that a RAC solution cannot run on most standard clouds. ScaleDB, on the other hand, runs over Ethernet, making it cloud compatible.

Degrees of Database Virtualization

There are a variety of ways to virtualize the database, each with its own collection of benefits. The following reviews some of the approaches to virtualization and exposes the degree of virtualization for each approach.

Running the Database in a Virtual Machine:
This simply means running a single instance of the database executable on top of a virtual machine. This enables you to release unused compute and storage to a pool of virtual resources, when the database is not fully utilizing them. When using a dedicated server, it is not uncommon for the database to only consume an average of 10% of the server's capacity. The ability to pool and repurpose these resources is a good first step toward database virtualization.

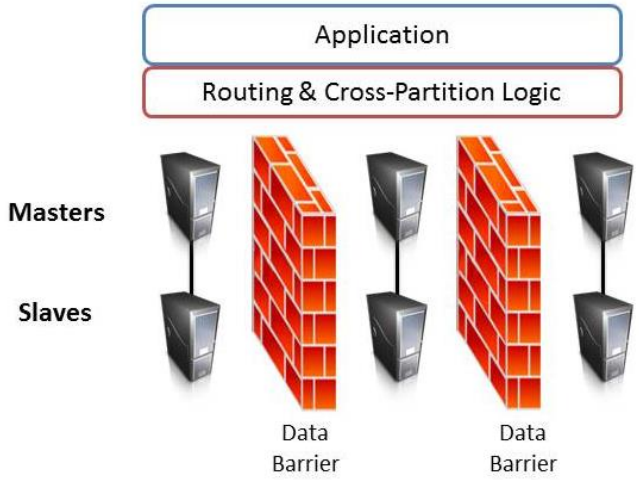
Another way of looking at this is allocating pooled resources *to* the database, as needed. For example, you can increase the memory, disk or CPU available for the database to use. This is nothing more than the inverse perspective or pooling unused resources—two sides of the same coin—but this is a perspective that is often used to describe the benefits of virtualization, earning it a mention here.

Since this approach still runs on a single instance of the database and since the compute and storage cannot scale independently, this approach offers only a limited benefit.

Virtualization and Sharding:
Sharding is one approach to partitioning the database, resulting in multiple identical images of the database, each storing different unique pieces of the data. For example if you have a million users, you might have ten shards containing 100,000 users each. Since each database has an identical schema, exceeding 1,000,000 users means you simply spin-up an eleventh image of the database.

The advantage of combining virtualization and sharding is that you can allocate resources on the fly to the virtual machines powering various shards, according to their needs. For example, if the shard containing users 400,001 – 500,000 uses less resources, you can reduce them accordingly. Similarly, if users 600,001 – 700,000 are heavy users of the database, you can allocate more resources to that shard.

Diagram 1 – Sharding creates database silos working independently of each other



While sharding and virtualization enjoy some synergy, sharding does not fully exploit the advantages of database virtualization. Sharding relies on a tight integration between the compute and the data files, so you cannot separate the two and scale them independently.

While each shard has a common schema, they are each limited to their unique piece of the data, and the application tier must know which piece (or shard) of that data is on which machine, so it can route the database requests accordingly. This creates physical silos of data that are tied to machines.

The downside of sharding is that the database shards all act as independent databases. This means that any function that operates across these shards must be moved from the database, where it is normally handled, into the application. For example, if you want to do joins, counts, range scans, aggregates, etc. that include more than a single shard, you have to code that capability in the application tier. This creates more work, introduces more potential bugs and is less efficient than simply processing requests in the database itself.

Additional Information: Wikipedia,
Tools: SQL Azure Federations, CodeFutures' dbshards, ScaleBase, various NoSQL Databases.

Storage Virtualization:
Databases typically address data as blocks, versus files, enabling them to operate on more granular chunks of data. In the past, Network Attached Storage (NAS) stored data as files, while Storage Attached Network (SAN) stored data as blocks, but now both NAS and SAN provide block storage. Leading vendors of both storage devices provide storage virtualization, effectively mapping logical requests for data to the physical location of the data. They both implement tiered storage plans, where most frequently used data is cached in memory, then solid state disks (SSD or Flash), an finally on rotating disks. By virtualizing the data, the most popular data can be moved, on the fly, from slower to faster storage media. This doesn't solve the issue of virtualizing the compute aspect of the database, but it does provide certain storage-centric advantages.

Additional Information: Wikipedia
Products/Services: Virtualized storage products are available from EMC, Netapp, HP, IBM, Hitachi, Dell, Oracle, Amazon and others.

Database Storage Virtualization/Replication:
The actual files or blocks of data stored by a database do not capture all of the "state" information of the database. There are also the transactions in process. Simply copying the files or blocks to another instance of the database fails to capture this state information, and yields an inconsistent copy of the data. In order to capture a consistent copy of the data, for use by another instance of the database, you must maintain the links between the databases. This can be accomplished by uni-directional replication (master-slave) or bi-directional replication (master-master)

Database Replication is built into most databases, and sends information—typically the log file—which is then "played" or executed by a slave database as if the commands were sent directly to that database.

Multi-Master Replication This is a bi-directional replication system that allows writes on more than a single master and then replicates those changes across a collection of servers. This is supported by various commercial databases or add-ons to those databases. More information about multi-master replication is available here. While multi-master sounds great, it is often a retro-fit or after-market fix to the database and can result in additional problems, including database inconsistency.

Data Replication does not cause the database to process a log, it creates and maintains the file- or block-level synchronization itself. Examples include Linbit's DRBD and Delphix. These approaches can be used to maintain a fail-over copy of the data, or to create copies for use in functions such as reporting/analytics, QA, test, development, etc.

Replicated In-Memory Databases:
In an effort to provide more mobility of the database instances, some companies have created in-memory databases By maintaining all of the data and state in RAM, it is well contained, fast and more mobile. In-memory DBMS have long been projected to replace disk-based database, but have perennially fallen short. While working in memory only provides performance advantages, relative to disk-based solutions, they have faced challenges with increasing data size, durability and recoverability. One of the earlier in-memory only databases was Times10, indicating a 10X performance advantage by remaining in memory. The most recent entrant to the in-memory database is SAP's Hana. VMWare has implemented in-memory databases in an effort to make them easier to virtualize [1, 2, 3].

Sharded Databases with SQL Routing:
When using a partitioned, or sharded, database you can run a SQL-aware load balancing process above the various sharded databases to facilitate routing database requests to the appropriate server. This SQL-aware load balancing process can be operated as a separate tool that works with various underlying databases (e.g. ScaleArc), or it can be all handled under the covers by extending the database itself (e.g. MySQL Cluster and Xeround).

By putting a router in front of a sharded or shared-nothing database, every database request requires two additional network hops, to the actual data and back again. If you have a single routing node, like Scalarc, you can include caching such as Memcached. However, this approach limits throughput, since you have only one node handling all routing, in order to avoid cache incoherency. Using a SQL-aware router in front of a standard database does not enhance availability of the database, it merely handles routing and/or caching.

The alternate approach—used by MySQL Cluster and Xeround—sacrifices the performance boost of local caching, in exchange for higher throughput by using multiple routing nodes. In order to

achieve reasonable routing performance, all indexes must be maintained in memory; otherwise, database requests could insert an additional disk look-up, which has a huge impact on performance.

Shared-Data Clustered Databases:
Oracle Real Application Clusters (RAC) and ScaleDB are shared-data clusters. Shared-data databases inherently separate the compute from the storage, enabling both functions to be virtualized and to scale independently of each other. By separating these two functions, and turning them into virtual building blocks, clusters can be assembled and modified on the fly, all without a single point of failure.

While some of the shared-data advantages mirror those provided by sharded databases with SQL routing, there are significant differences under the covers that manifest themselves in a superior performance profile. The performance advantages are primarily in the following areas:

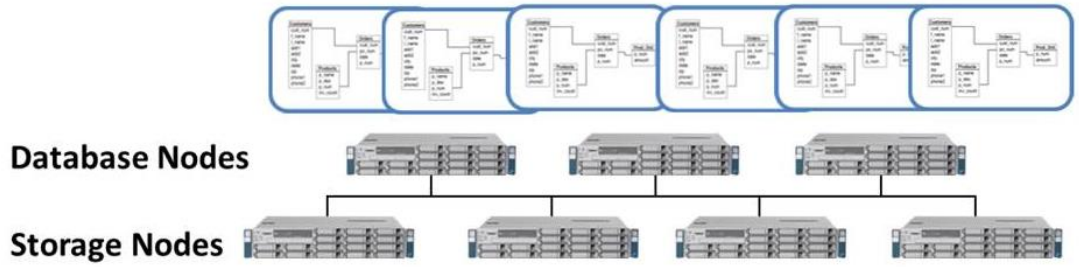
1. Performance improvement from local caching on the first database node contacted, which frequently avoids additional network hops.

2. Faster cross database functions (e.g. joins, range scans, counts, aggregates, etc.) since each database node see the entire database and can process any function by itself, without involving other shards.

3. Indices are not limited to memory, and can overflow into disk, while still delivering excellent performance.

4. Queries can be distributed across the smart storages, in parallel, to further boost performance. This is analogous to map reduce, where each storage node processes its portion of the data, and then the database node combines the results from various storage nodes. For example, if the database gets a request for all sales in the past week, this request can be sent to the storage nodes, which process their portion of the data locally and only send the results. This both parallelizes the processing and reduces the network traffic, since it only send results over the network and not the entire table.

Diagram 2 – Shared-data DBMS runs multiple virtual instances of the database on virtual machines



By separating and virtualizing the compute and storage functions of the database, shared-data DBMS deliver more of the benefits of database virtualization than any other approach.

Conclusion

Virtualization, by virtue of the rich benefits it delivers, has been an unstoppable force in the server market. Cloud computing has accelerated the adoption wave. The success of server virtualization has led to virtualization of storage and now networking. In both of these cases the benefits to IT personnel and to users have been overwhelming.

At this point, all aspects of the typical cloud environment can be completely virtualized with the sole exception of SQL databases. ScaleDB brings the shared-data architecture to MySQL, the most popular database in the cloud today. This shared-data architecture converts MySQL from being unvirtualizable into being a completely virtualized database.