ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 10, October 2014

The Use and Industrial Importance of Virtual Databases

Dr V S Dhaka, Sonali Vyas

Professor & Head, Dept of CSE, Jaipur National University, Jaipur, India

Research Scholar, Dept of CSE, Jaipur National University, Jaipur, India

ABSTRACT: Virtualization refers to the abstraction of logical resources away from their underlying physical resources to improve agility and flexibility, reduce costs, and thus enhance business value. Virtualization allows a set of underutilized physical infrastructure components to be consolidated into a smaller number of better utilized devices, contributing to significant cost savings. 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. While SQL databases are the most popular for addressing 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 the 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 ideally suited to be virtualized.

KEYWORDS*:* Virtualization, Shrading, DV, Query Complexity, Data Replication

I. INTRODUCTION

Virtualization is the creation of a virtual version of something (operating system, storage devices, database, 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, that server 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, allocating resources to virtual machines can be done dynamically, enabling a greater degree of flexibility and more granular management.

II. LITERATURE REVIEW OF 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.

2.1 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[1-3]:

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 10, October 2014

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].

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[4].

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[5, 6].

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

2.2 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 computers 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.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 10, October 2014

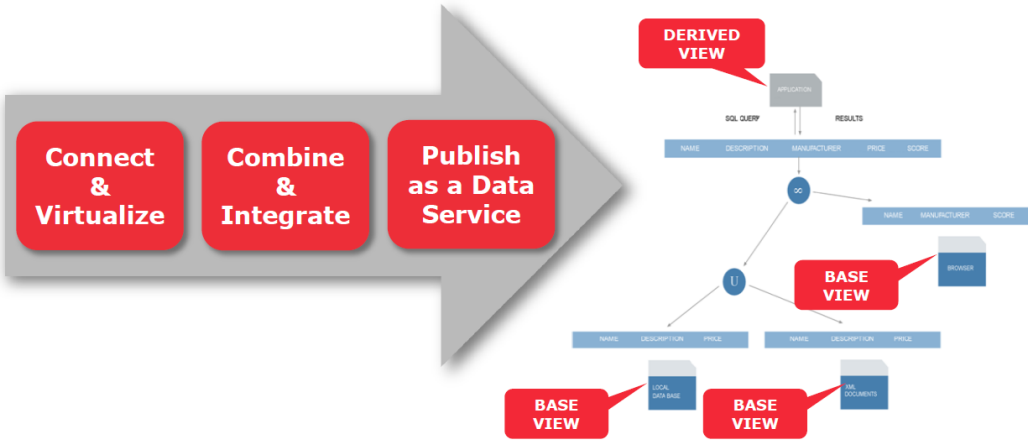

Figure 1: Basic Understanding of Database Virtualization

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.
Data virtualization, DV for short, attempts to perform data cleansing, data transformation and data correlation as data moves out from production systems thus avoiding any intermediate storage [9]. This is opposed to Data warehouse approach which physically changes data in each stage and loads it in to some data store [10]. Typical Data virtualization platform requires:

- Ability to Discover data stored in data sources
- Ability to retrieve data from different data sources
- Ability to define views or virtual tables
- Ability to optimize federated query
- Ability cache data
- Fine grained security

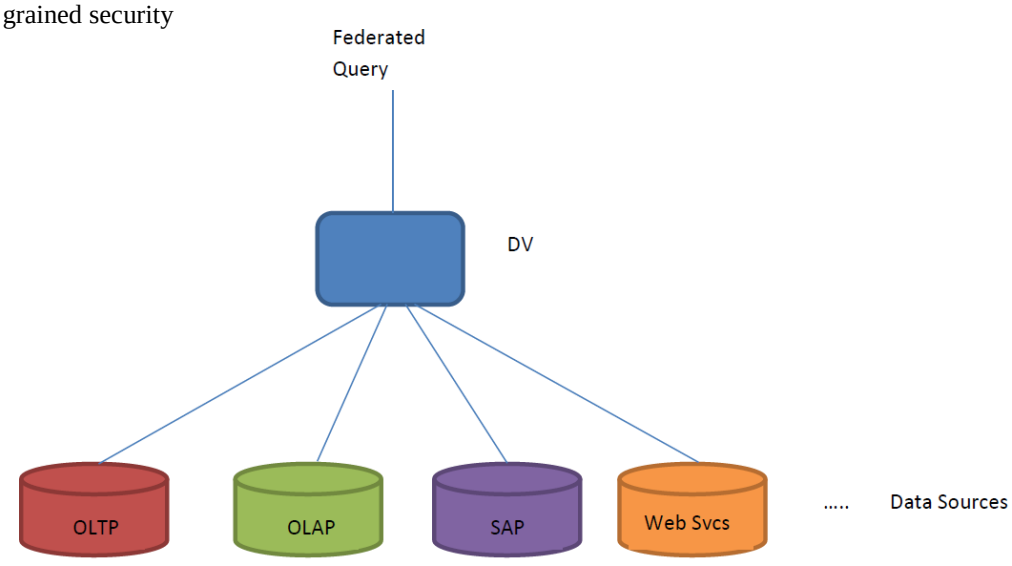

Figure 2: Data Virtualization – Overview

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 10, October 2014

The above diagram depicts usage of DV platform for integrating data from databases, SAP and web services. Each phase of data analysis gets defined using virtual tables; please refer to section below on more details on virtual tables. Each phase uses data from previous phase by referring to virtual table from previous section. When analysis needs to be done DV compiles all of the definitions of virtual tables in to a single SQL, which is then compiled, optimized and executed.

2.3 Data Source Connectors

DV platform needs to bring data from disparate data sources. Different data sources have different access mechanisms. Data coming from data bases would need queries submitted through SQL using JDBC/ODBC connections, data coming from supply chain and financial apps would require their proprietary access mechanisms, fetching data from web services would require web service access methods. DV platform needs to have data access mechanisms for all these different data sources [11, 18].

2.4 Data Discovery

Data may need to be retrieved from variety of data sources like databases, applications, flat files, web services. DV platform needs to understand the schema of data storage and relationship among them. For example with data source that uses data base, a DV platform needs to figure out all of different tables in the database, constraints defined on them, primary-key, foreign key relationships. Databases typically store this information in their system catalogs which could be used by DV[13]. The metadata about these schemas needs to be stored with in DV platform so that when user submits a query, DV platform can properly execute the query by fetching data appropriately.

2.5 Virtual Tables/Views

Virtual Tables or Views are the result set of a stored query which can be used just like a regular database table. The table schema and table contents are defined by SQL. The table is considered virtual because table contents are not physically stored. Data for the virtual table is brought in from underlying database tables when query defining virtual table is executed.

For example “Inventory_Data” is a virtual table that has columns “productid”, “inventory” and “cost”. The contents of the table “Inventory_Data” comes from database table “Inventory_Transactions”.

View Inventory_Data:

Select productid, inventory, currencyConvertToDollars(cost) as cost from Inventory_Transactions

Inventory_Transactions Table in DataBase

ProductID	Inventory	Cost	BackLog	Supplier
10	100	5000		XYZ Tech

In order to be used for data integration DV platform must do data cleansing, data transformation and data correlation. DW does each of these stages separately and produces physically transformed data at each stage. Unlike DW and ETL, DV platform would do all of these stages mostly in one step before producing the final report [16]. DV platform needs to define data cleansing, data transformation and data correlation logic programmatically using SQL like query language. Typically this is achieved by defining views or virtual tables; typically users would define multiple such views at different levels of abstraction. When a report is generated data moves from data sources through the layers of these views cleansing, transforming, joining data before producing the report.

In some cases DV may use implicit virtual tables to expose data from data sources. This is because underlying data source may not be relational and instead could be hierarchical, multi-dimensional, key value store, and object data model; DV needs to convert all of these data models back in to relational data model and provide SQL as the language for data access. In such cases DV would expose underlying data as tables; for such implicit virtual tables, there wouldn’t be any SQL defining the schema instead it would be natural mapping of underlying data in to relational able[14, 17].

.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 10, October 2014

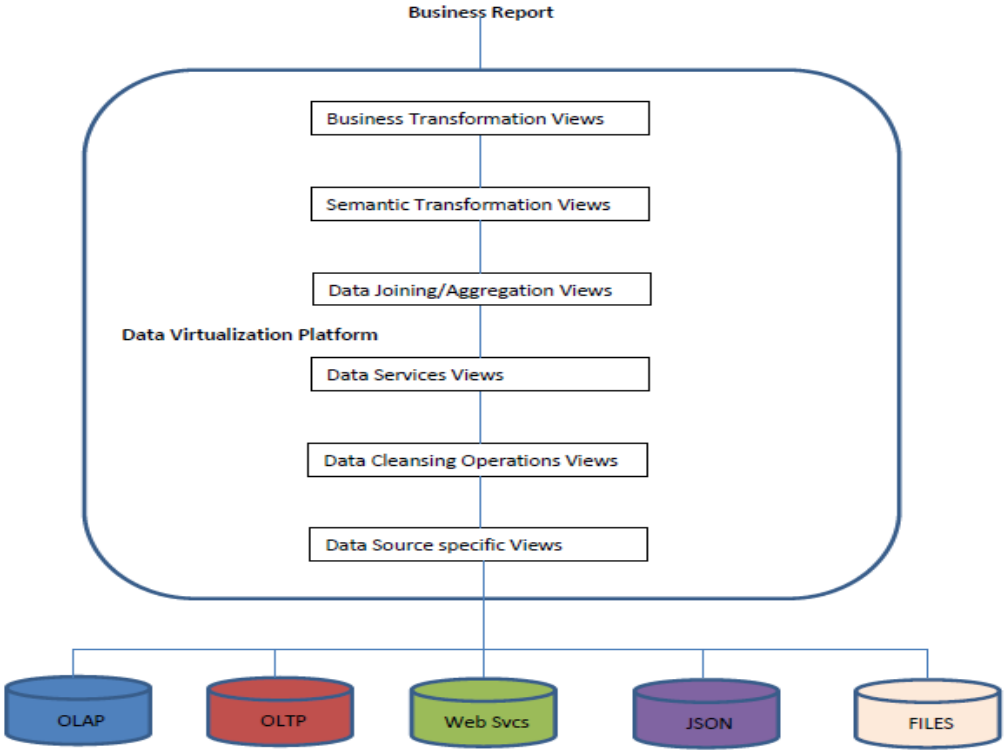

Figure 3: Data Virtualization – Example of Virtual TablesData Virtualization – Example of Virtual Tables

The above diagram depicts deployment architecture for data integration using DV. In this example there are six different types of Virtual Tables one using the other as the source table. Data of interest from each data sources is first defined by data source specific views. Data cleansing views takes data from data source specific views and cleans them of data errors. Typically cleansing of data entry errors is done by passing data through custom build procedures. Transformation for semantic differences may be defined as separate virtual tables. Cleansed data is exposed as data services for each business entity. Data from the various business entities is correlated which is then transformed as required by analysis use cases. This data may optionally further go through additional transforms.

At each stage data from previous stage is accessed by specifying the virtual table from previous stage as the source of data. When a query is submitted to the top level virtual table (Business Transformation Views), DV would assemble the complete SQL by chaining user submitted query with the SQL that defines virtual tables from all stages [20].

III. COST BASED QUERY OPTIMIZER

Since data cleansing, data joining, data transformation is all defined in virtual tables and since data is fetched from many different data sources, the relevance of optimal query execution becomes very important. Figuring out the cost of bringing data from different data sources and then planning query execution would reduce the query latency. Cost based optimizer is an important feature in reducing query latency time.

In the traditional data integration approach since the whole data is already in DW, the cost of fetching data from different data sources is avoided. Also DW often would have indexes built on data. Since DV doesn't store data, DV cannot take advantage of indexes. Most DW databases would have cost based optimizers, but focus of DW optimizer is to cut disk access time whereas in DV the focus of optimizer is to push work load as much as possible to backend production systems.

IV. DATA CACHING

Ideally all of the data cleansing, joining and transformation should happen as data flows out from production systems; however this approach has some side effects:

- Query Latency may increase
- May impose load on production data stores
- May loose historical data

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 10, October 2014

To avoid these problems sometimes it may be desirable to load intermediately transformed data in to data stores; this is effectively data caching. For example the output of data cleansing may be cached to avoid the load on production systems. Caching data needs to take in to account the rate of change of data in data sources.

V. FINE GRAINED SECURITY

Since DV platform needs to bring data from different data sources, it needs to manage the authentication and authorization to access various data sources. Similarly DV platform needs to return data to users based on their credentials. This would often require data masking and data filtering based on user authorization.
ETL process also needs to access data from production systems. Based on whether pull or push model is employed ETL needs to manage authentication and authorization to back end production systems. Similarly DW needs to deliver data based on authorization. In terms of security requirements there is actually not much difference between DV and ETL-DW.

VI. 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[21-23]. 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- disk 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-disk, 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 is 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 one has implemented a shared-data database architecture, he has solved all of the virtualization challenges; and 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

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 10, October 2014

that a RAC solution cannot run on most standard clouds. ScaleDB, on the other hand, runs over Ethernet, making it cloud compatible.

VII. 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.

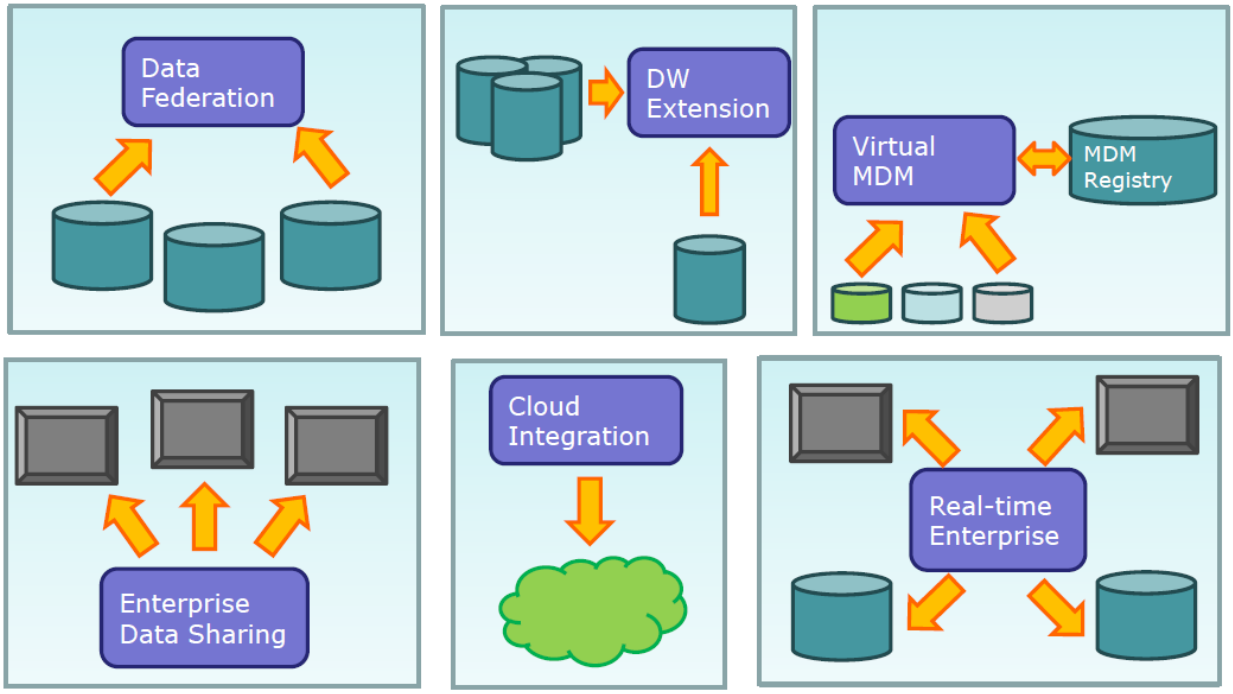

Figure 4: Running the Database in a Virtual Machine:

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[23]. 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.

VIII. 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.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 10, October 2014

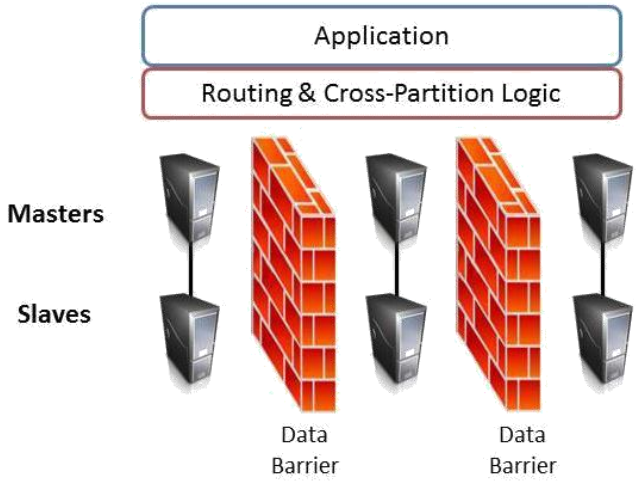

Figure 5: Sharding creates database silos working independently of each other

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 [25]. 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. Tools: SQL Azure Federations, CodeFutures' dbshards, ScaleBase, various NoSQL Databases.

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

9.1 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

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 10, October 2014

"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.

9.2 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].

9.3 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.

9.4 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

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 10, October 2014

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.
5. This both parallelizes the processing and reduces the network traffic, since it only send results over the network and not the entire table.

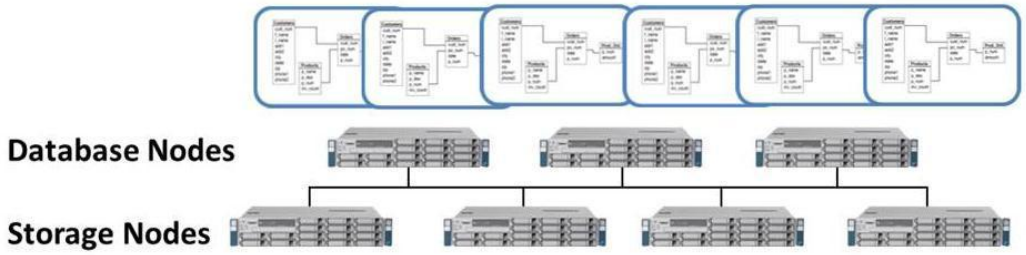

Figure 6: Shared-disk DBMS runs multiple virtual instances of the database on virtual machines

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.

X. CONCLUSION

In this paper we have presented an extensive study of different database virtualization techniques, their advantages and disadvantages along with trending market scenario. Virtualization, by virtue of the rich benefits it delivers, has been an unstoppable force in the server market. The detailed study presented in this paper concludes that Database Virtualization has great importance and benefits in remote and underlying architecture independent data access. Many organizations have already adopted DV and are reaping benefits from it. Some of these use cases are depicted in "Data Virtualization" book, printed Sep 2011, by Judith R. Davis and Robert Eve. Data Virtualization software companies like Composite Software, Denodo, Informatica and IBM seems to be growing their Data Virtualization business. Though the database virtualization is very useful technique but it have some limitations as well. The research community has to work extensively to overcome these limitations. In other words these are the future work directions for the researchers.

REFERENCES

[1] Figueiredo R, Kapadia N and Fortes J. A. B., "The PUNCH Virtual File System: Seamless Access to Decentralized Storage Services in a Computational Grid", Proc. IEEE International Symposium on High Performance Distributed Computing (HPDC), August 2001.
[2] Backman V, Gurjar R, Badizadegan K, I. Itzkan, R. R. Dasari, L. T. Perelman, M. S. Feld, "Polarized light scattering spectroscopy for quantitative measurement of epithelial cellular structures", IEEE J Sel Top Quant. Elec., 5, 1019, 1999.
[3] Backman V, et al. "Detection of preinvasive cancer cells in situ", Nature, 406, 35-36, 2000.
[4] Backman V, Gurjar R, Perelman LT, Georgakoudi I, Badizadegan K, Itzkan I, Dasari RR, Feld MS, "Imaging human epithelial properties with polarized light-scattering spectroscopy", Nature Medicine, 7, 1245-1248, 2001).
[5] Gvon Laszewski et al., "Real-time Analysis, Visualization, and Steering of Tomography Experiments at Photon Sources", Proc. 9th SIAM Conf. on Parallel Processing for Scientific Computing, Apr 1999.
[6] Smallen S., Casanova H. and F. Berman, "Applying Scheduling and Tuning to On-line Parallel Tomography", Prof. of Supercomputing, Denver, Nov 2001.
[7] Smallen S. et al., "Combining Workstations and Supercomputers to Support Grid Applications: The Parallel Tomography Experience", 9th Heterogeneous Computing Workshop, May 2000.
[8] Apostolico A. et al, "Requirements for Grid-Aware Biology Applications", DataGrid WP10 Workshop, DataGrid-10-D10.1-0102-3-8, Sept 2001, http://marianne.in2p3.fr/datagrid/wp10

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 10, October 2014

[9] Chervenak, I. Foster, C. Kesselmann, C. Salisbury, S. Tuecke, "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets", to appear, Journal of Network and Computer Applications, 23(3) p187-200 July 2000.

[10] Hoschek W., Jaen-Martinez J., Samar A., Stockinger H. and K. Stockinger, "Data Management in an International Data Grid Project", IEEE/ACM Intl. Workshop on Grid Computing (Grid'2000), Dec. 2000.

[11] Plank J, Beck M., Elwasif W., Moore T., Swany M. and Wolski R., "The Internet Backplane Protocol: Storage in the Network", Network Storage Symposium (NetStore), Seattle, WA 1999.

[12] Bester J., FosterI., Kesselman C., Tedesco J. and Tuecke S., "GASS: A Data Movement and Access Service for Wide Area Computing Systems", Proc. 6th Workshop on I/O in Parallel and Distributed Systems, May 1999.

[13] Henderson R. and Tweten D., "Portable Batch System: Requirement Specification", Technical Report, NAS Systems Division, NASA Ames Research Center, Aug. 1998.

[14] Litzkow M., Livny and Mutka M. W., "Condor: a Hunter of Idle Workstations", Proc. 8th Int. Conf. on Distributed Computing Systems, pp104-111, June 1988.

[15] Thain D., Basney J., S-C. Son, and Livny M, "The Kangaroo Approach to Data Movement on the Grid", Proc. 10th Intl. Symp. on High Performance Distributed Computing (HPDC), pp325-333, Aug. 2001.

[16] White, Grimshaw A, and Nguyen-Tuong, "Gridbased File Access: the Legion I/O Model", in Proc. 9th IEEE Int. Symp. on High Performance Distributed Computing (HPDC), pp165-173, Aug 2000.

[17] White, Walker M, Humphrey M., Grimshaw A., "LegionFS: A Secure and Scalable File System Supporting Cross-Domain High-Performance Applications", Proceedings of Supercomputing (SC), Nov 2001.

[18] Allcock A, Bester J., Bresnahan J., Chervenak A., Foster I., Kesselman C., S. Meder, V. Nefedova, D. Quesnel, S. Tuecke. Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing, IEEE Mass Storage Conference, 2001.

[19] Callaghan B., NFS Illustrated, Addison-Wesley, ISBN 0-201-32570-5, 2002.

[20] Morris J., Satyanarayanan M., Conner M., Howard J., D. Rosenthal and F. Smith, "Andrew: A Distributed Personal Computing Environment", Communications of the ACM, 29(3) pp184-201, March 1986

[21] M. Satyanarayanan, J. Kistler, P. Kumar, M. Okasaki, E. Siegel, D. Steere, "Coda: A Highly Available File System for a Distributed Workstation Environment", IEEE Transactions on Computers, 1990, 39(4), 447-459.

[22] S. Adabala, V. Chadha, P. Chawla, R. Figueiredo, J. Fortes, I. Krsul, A. Matsunaga, M. Tsugawa, J. Zhang, M. Zhao, L. Zhu, and X. Zhu. "From Virtualized Resources to Virtual Computing Grids: The In-VIGO System", to appear, Future Generation Computing Systems, special issue, Complex Problem-Solving Environments for Grid Computing, David Walker and Elias Houstis, Editors.

[23] Figueiredo R, Dinda P. and Fortes J., "A Case for Grid Computing on Virtual Machines", Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS), May 2003

[24] Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. Lam and M. Rosenblum, "Optimizing the Migration of Virtual Computers", Proceedings of the 5th Symposium on Operating Systems Design and Implementation, 2002.

[25] Figueiredo R. J., "VP/GFS: An Architecture for Virtual Private Grid File Systems". In Technical Report TR-ACIS- 03-001, ACIS Laboratory, Department of Electrical and Computer Engineering, University of Florida, 05/2003.