Shared-Nothing vs. Shared-Disk Cloud Database Architecture

Sunguk Lee

Research Institute of Industrial Science and Technology
Pohang,Korea
sunguk@rist.re.kr

Abstract

The cloud computing is the new promising computing system. It is evident that storage plays a major part in the data center and for cloud services. The storage virtualization plays a key part in the dynamic infrastructure attribute of Cloud Computing. Which means the storage is provisioned and de-allocated on demand and usage needs. The cloud protocol, architecture, implementation and services specifications are still in its immature stage. One of these specifications is the database architecture that is being used. There are two database architecture, the shared-disk and the shared-nothing. The primary database architectures, shared-disk and shared-nothing, each have their advantages. In this paper we discussed the characteristic, advantages and disadvantages of both database architectures for cloud computing.

Keywords: *Cloud computing, shared-disk, shared-nothing, database*

1. Introduction

Currently Cloud platforms have very little support for database design related virtualization enhancements. But in future designing databases specific for Cloud especially for private clouds in large enterprises is a sure possibility. In this context the following design principles are important when you design database applications which need to be delivered using Cloud platform [2].

It is evident that storage plays a major part in the data center and for cloud services. The storage virtualization plays a key part in the dynamic infrastructure attribute of Cloud Computing. Which means the storage is provisioned and de-allocated on demand and usage needs. The good part is that this complex stuff is hidden from the cloud consumer. However while the storage allocation is abstracted it also brings in performance concerns in a multi-tenant cloud environment where by most of the cloud consumers are geographically dispersed and a good amount of data retrieval and manipulation at stake.

Databases played an important role in any Information System. In cloud computing, there have been a lot of issues about the shared nothing and shared disk database design. Many researches, articles and blogs outlined the advantages and disadvantages of both approach. But what is really the real score between these two database designs?

The two primary DBMS architectures are shared-nothing and shared-disk. Shared-nothing databases split or partition the data so that each database server exclusively processes and maintains its own piece of the database. Shared-disk is analogous to a single large trough of data, where any number of database nodes can process any portion of that data. Based upon traditional computing constraints, the shared-nothing architecture has been the price-performance leader [3].

Shared-disk and shared-nothing represent forms of data access architectures. In a shared-disk resource model various processes in the DBMS have access to all system resources, including the data. In the shared-nothing environment, separate DBMS resources divide up

the workload, responsible for its own data, memory locations, and other resource [2].

In this paper we discussed the background of the shared-nothing and shared-disk, what are the advantages and disadvantages of both. And finally make our conclusion based on the presented data that we got after going through many researches.

2. Shared-Nothing

In Shared-Nothing environment each system has its own private memory and one or more disks. The clustered processors communicate by passing messages through a network that interconnects the computers. In addition, requests from clients are automatically routed to the system that owns the resource. Only one of the clustered systems can "own" and access a particular resource at a time. Of course, in the event of a failure, resource ownership may be dynamically transferred to another system in the cluster. Shared-nothing databases use the local disk, while shared-disk databases rely on storage that is shared and accessible via the network. Because of this, shared-nothing databases had an inherent performance advantage in an era of slow networking [3].

The main advantage of shared-nothing clustering is scalability. In theory, a shared-nothing multiprocessor can scale up to thousands of processors because they do not interfere with one another, nothing is shared. For this reason shared-nothing is generally preferable to other forms of clustering. Furthermore, the scalability of shared-nothing clustering makes it ideal for read intensive analytical processing typical of data warehouses.

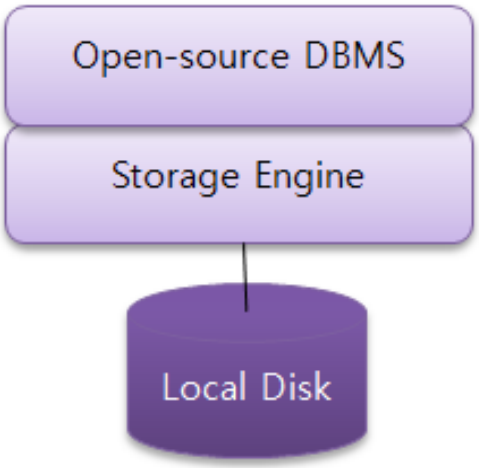

Figure 1. Shared-Nothing

Figure 1 illustrates the shared-nothing database architecture, where it has its own private memory and one or more disks.

As evident implementing a Shared-Nothing Cloud Database Model does not replace Storage Virtualization, but compliments the scalability further by splitting the data within multiple virtual storage drives.

Because it lacks dynamic load balancing, shared-nothing requires that each server accommodate the peak load for the data it owns. This causes shared-nothing implementations to invest more in servers using either a scale-up or a scale-out approach. Because of latency issues between geographically distributed databases, you should partition data based upon geo-location to minimize or eliminate function- and data-shipping between disparate locations.

3. Shared- Disk

Shared-Disk environment all of the connected systems share the same disk devices. So each processor still has its own private memory, but all the processors can directly address all the disks. Typically, shared-disk clustering tends not to scale as well as shared-nothing for smaller machines. But with some optimization techniques shared-disk is well-suited to larger enterprise processing, such as is done in the mainframe environment. In particular, the coupling facility and database robust optimization technology helps to enable efficient shared-disk clustering. Mainframes are already very large processors capable of processing enormous volumes of work. Great benefits can be obtained with only a few clustered mainframes, whereas many PC and midrange processors would need to be clustered to achieve similar benefits. Shared-disk usually is viable for applications and services requiring only modest shared access to data as well as applications or workloads that are very difficult to partition. Applications with heavy data update requirements probably are better implemented as shared-nothing.

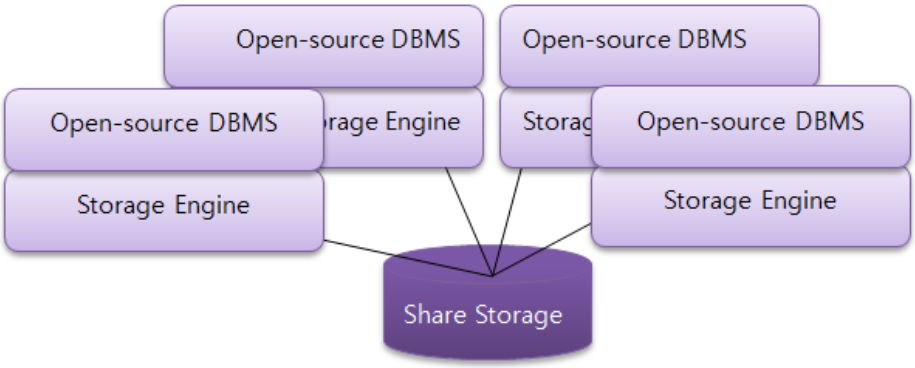

Figure 2. Shared- Disk

Shared-disk enables any node to access the entire data set, so any node can service any database request. These results in more fluid load-balancing than with shared-nothing. This fluidity is the driving factor behind shared-disk's ability to smoothly accommodate temporal and evolutionary changes in usage patterns. It also enables you to run each server at a higher CPU utilization, because peak loads are spread across all of the servers in the cluster. Shared-disk has the same latency issues between geographically remote locations as shared-nothing. Address this situation by building two separate shared-disk systems, in effect partitioning them, like you would with a shared-nothing database.

4. Comparisons in Scalability, High-Availability, Load Balancing and Data Consistency

4.1 Scalability

Scalability is the measure of a system's ability respond to increased or decreased workload with minimal, or no manual intervention required. When load exceeds the capabilities of the systems in the cluster, additional systems can be added to the cluster

Shared-nothing, depending upon your partitioning scheme, function- and data-shipping volume can seriously limit your scalability. For websites in search of hyper scale, partitioning is scalable and very popular. Partitioning is ideal for read-only or read-centric solutions that don't involve multi-node transactions.

Shared-disk also has its scaling challenges, but these challenges are driven largely by the inter-nodal messaging. Shared-disk databases rely upon a message or token-passing model,

where the messages alert the nodes to the status of the other nodes. The more nodes you add to the system, the longer it takes to pass this token, which results in longer wait-states. The impact of this on scalability is the single biggest knock against shared-disk databases.

4.2 High-Availability

High-Availability clusters support server applications that can be reliably utilized with a minimum of down-time. They operate by harnessing redundant clusters that provide continued service when system components fail.

The shared-nothing architecture relies on a collection of single points of failure, master nodes, which are typically supported by one or more slaves per master. These slaves are waiting for the master node to fail, at which point they must be promoted to master status. This promotion is typically a manual process that can take several minutes to hours, once the alert has been received. The newly promoted master must also apply any uncommitted transactions in its log. The application must also know how to reroute the database calls to this new master [4].

Shared-disk clusters respond to server failure by routing database requests to the next available node. There is no need to promote servers or alert the application or data layers to re-route database requests. The failure of a server, in a shared-disk database, doesn't result in a fire drill. Instead the load is spread over the remaining servers in the cluster, while the DBA revives the troubled node at his leisure [4].

4.3 Load Balancing

Load Balancing is a technique to spread work between two or more computers, network links, CPUs, hard drives, or other resources, in order to get optimal resource utilization, throughput, or response time

In Shared-Nothing, because it lacks dynamic load balancing, shared-nothing requires that each server accommodate the peak load for the data it owns. This causes shared-nothing implementations to invest more in servers using either a scale-up or a scale-out approach [4].

Shared-disk enables any node to access the entire data set, so any node can service any database request. These results in more fluid load-balancing than with shared-nothing. This fluidity is the driving factor behind shared-disk's ability to smoothly accommodate temporal and evolutionary changes in usage patterns. It also enables you to run each server at a higher CPU utilization, because peak loads are spread across all of the servers in the cluster. [4].

4.4. Data Consistency

Data consistency, very simply means that the information you receive is valid and accurate across your entire cluster. There are two ways databases can encounter data consistency: First, a node reads the "dirty" data either in the middle of a change by another node or as a result of a failure by a node. Second, as a result of time delays for data replication, a copy of the data is out of date, resulting in one node giving one answer and another giving a different answer at the same time. These two issues might sound arcane and trivial, but that depends upon your business requirements.

In Shared-Nothing, you have three options: first, live with inconsistent data; second, configure your slave servers not to provide read access and reduce read performance; and last, configure your transactions to include synchronous replication inside the transaction, thus reducing overall performance. While Shared-Disk Data consistency is inherent because there is only one copy of the data [4].

5. Type of Clustering

Typically, shared-disk clustering tends not to scale as well as shared-nothing for smaller machines. But with some optimization techniques shared-disk is well-suited to larger enterprise processing, such as is done in the mainframe environment. In particular, the coupling facility and DB2's robust optimization technology helps to enable efficient shared-disk clustering. Mainframes are already very large processors capable of processing enormous volumes of work. Great benefits can be obtained with only a few clustered mainframes, whereas many PC and midrange processors would need to be clustered to achieve similar benefits.

Shared disk usually is viable for applications and services requiring only modest shared access to data as well as applications or workloads that are very difficult to partition. Applications with heavy data update requirements probably are better implemented as shared-nothing. Consult Table 1 for a summary of the capabilities of shared-disk versus shared-nothing clustering.

Table 1. Summary of the Capabilities of Shared-disk versus Shared-nothing Clustering

Shared-Disk	Shared-Nothing
Quick adaptability to changing workloads	Can exploit simpler, cheaper hardware
High availability	Works well in a high-volume, read-write environment
Dynamic load balancing	Fixed load balancing based upon the partitioning scheme
Performs best in a heavy read environment	Almost unlimited scalability
Data need not be partitioned	Data is partitioned across the cluster
Messaging overhead limits total number of nodes	Depends on partitioning, data shipping can kill scalability

6. Conclusions

The debate between the shared-disk and shared-nothing database architecture is now topic of many developers. In this paper we presented the different views in both shared-disk and shared-nothing architecture in terms of scalability, high-availability and load balancing. Also, we discussed about the advantages and disadvantages of both architecture. It is important to note, what are the advantages and disadvantages of both architecture and what is best suited to the needs and requirements of the cloud database for enterprise, developer and costumer after considering the advantages and disadvantages of both architectures.

References

[1] A. Jhingran, "Shared Nothing vs. Shared Disks, the Cloud Sequel?", http://jhingran.typepad.com/anant_jhingrans_musings/2010/02/shared-nothing-vs-shared-disks-the-cloud-sequel.html, Retrieved, 2012/7/23.
[2] S. SundaraRajan, "Cloud Database Design, Scale Out Using Shared Nothing Pattern", http://cloudcomputing.sys-con.com/node/1586119, Retrieved 2012/7/23.
[3] M. Hogan, "Database Virtualization and the Cloud", Cloud Database White paper, http://www.scaledb.com/pdfs/Cloud_Databases_WhitePaper2.pdf, Retrieved 2012/7/23.
[4] M. Hogan, "Shared-Disk vs. Shared-Nothing Comparing Architectures for Clustered Databases", http://www.scaledb.com/pdfs/WP_SDvSN.pdf, Retrieved 2012/7/23.