International Journal of Information and Computation Technology.
ISSN 0974-2239 Volume 3, Number 7 (2013), pp. 701-710

http://www. irphouse.com /ijict.htm

Reincarnating Traditional Relational Database through NoSQL

Renu Kanwar[1]and Prakriti Trivedi[2]

[1]Govt. Engineering College, Ajmer, Rajasthan, India.
[2]Govt. engineering college, Ajmer, Rajasthan, India.

Abstract

Day by day data is getting bigger and bigger we need to increase the mode of data storage as well, a common problem encountered with our traditional relational database is the problem of data bottleneck which still exists with the traditional methods of data storage this gave birth to NoSQL databases, commonly referred as "Not only Structured Query Language" which modifies some of the parameters of the existing databases and removes it's limitations. NoSQL do not make use of the basic protocols of RDBMS like ACID, instead works on BASE which promises eventual consistency. Introducing Coherence which is the a product of oracle works on NoSQL distributing data in such a fashion that small sized data are replicated and big data are distributed using facts and dimensions which makes scaling easy and increases the speed of system enormously, Thus we can say that NoSQL works according to the designer instead of designer working according to database.

Index Terms: NoSQL; Data-bottleneck; Coherence; facts; dimensions.

1. Introduction

Considering today's era and the unsullied progress in cloud computing there are various use cases where the traditional relational database limits themselves [4], considering the following facet, like where the Big Data Storage is concerned large application like search engines and popular social networking sites require enormous amount of data to be stored and respond well with huge data traffics, Speed and Scalability for high performance of the system when concurrent read/writes are

involved with flexibility to scale up the storage area as per the requirement without increasing any additional overheads[2], High availability and fault tolerance for the same system should be readily available all the time without any single point of failure, if any server is disconnected then also the task assigned to that server should go on, it should provide fast data backup and recovery. Thus it can be said that we require applications which are highly scalable and could modify according to the designer's wish. There are assortments of challenges faced by the relational database (RDBMS) which make it vulnerable to changes.

The basic problem faced by the relational databases today is the problem of data bottlenecking which arises when there is large number of concurrent requests for a single application and system fails to respond to all the requests simultaneously. To solve this problem with database data clustering was done where the same data was replicated to multiple sites to avoid bottlenecking at particular site but this solution was not appropriate as it faced the data synchronization problem and hence the problem persisted.

Traditional RDBMS possesses a database schema which consists of a too many joins and relationships which makes the data storage a complex issue and ACID property increases the overall overheads there are some of the instructions like buffer manager, hand code optimization etc which are carried out unnecessarily and further adds up to the complexity[10].

To condense the problems faced by traditional relational database NoSQL came into existence as it does not work up on the basic principles of SQL therefore do not have joins and complicated relationships, NoSQL uses BASE instead of using strict ACID constraints where BASE stands for Basically available, Soft state, Eventual Consistency [1]which means while working the system is not responsible to maintain consistency at each state rather the belief is on making the system consistent at the end of any process and clients faces an inconsistent state of data while updates are in progress..

2. NoSQL Functioning

NoSQL is regarded as a revolution in the field of grid computing and whenever distributed databases are discussed as it liberate us from all the setbacks of RDBMS like storing hefty data, blazing speed, extensive scalability, accessibility and excellent backups.

NoSql works to achieve the shared nothing architecture which is difficult to accomplish by RDBMS , it is required to achieve this kind of architecture because RDBMS faces a predominant problem of data bottlenecking as it have a shared architecture shown in Fig 1, for which a solution database clustering is present, but it could be done within limits as too much data replication will eats up lot of storage space and problem of distributed locking will arise, another method is the database sharding as in fig2, where the database is divided into two or more parts to avoid distributed locking but here data synchronization becomes another issue thus we focus

on achieving a totally shared nothing architecture , the term shared nothing means that when the data repository is sharded or divided the database should not be dependent on each other for any kind of manipulations and any updates inside any database should not affect the other.

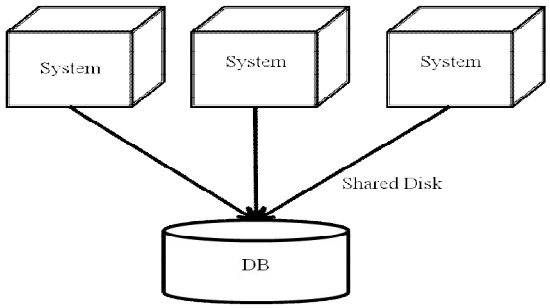

Figure 1: Shared architecture

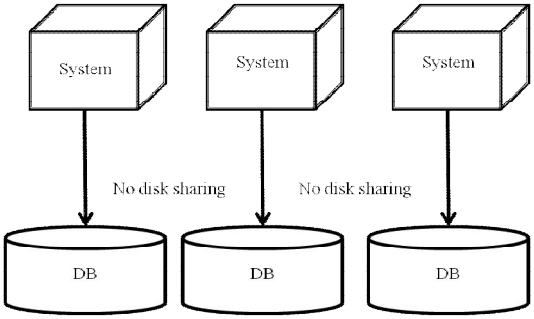

Figure 2: Shared Nothing architecture

Shared nothing architecture can be achieved by introducing a scalable state which lies in the application layer and data is stored inside the memory RAM which is much more quick as compared to the disk[7], and this added extra layer as in fig 3 could be scaled out very easily and as data could be cached from the RAM the disk access time is reduced.

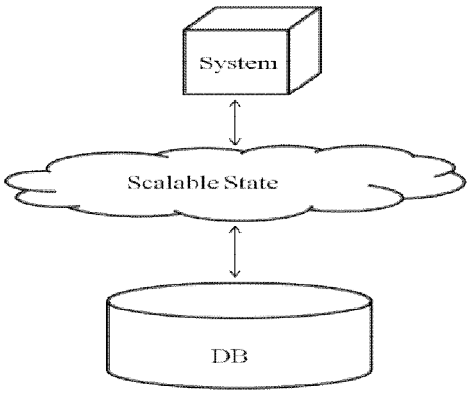

Figure 3: NoSQL architecture.

But the major problem with the In-memory data storage is that it is fragile and data could be lost. Thus we need to place the modified data finally in the major data repository asynchronously like data could finally reach to a persistent data store when the data traffic is idle possible during late nights.

3. Coherence

A fine product of oracle successfully running in various multinationals it basically works as a cache and offers some more functions as query, functionality, indexing etc [9]. Its architecture consists of clustered nodes connected to each other and behind there is a persistent data store shown in fig 4.

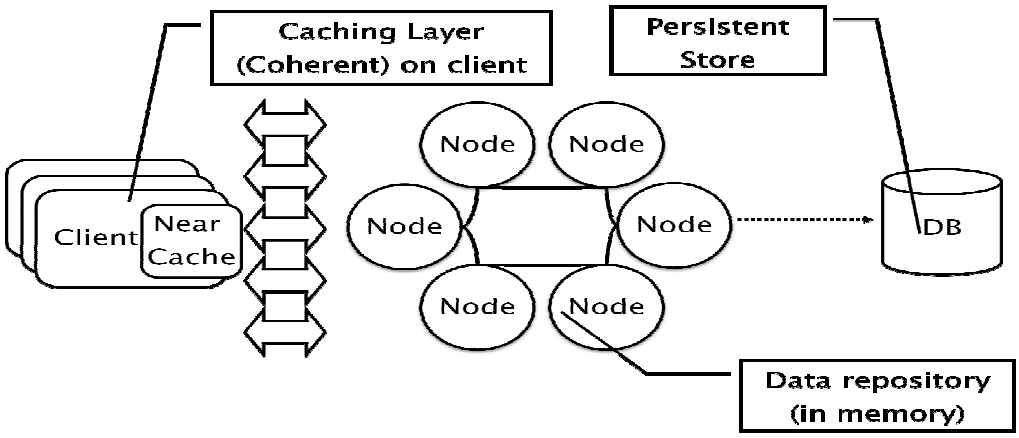

Figure 4: Coherence Architecture.

There are basically three well-defined layers [9]. First is the client second the clustered node and third persistent data store, when the client request for a data, the requested data is present in the clustered node and it is made available to client after all the required modifications it is again stored in the data repository also if any data not in the clustered node is populated from the data store, there are basic three features of Coherence like it is fast, fault tolerant and highly scalable. The backup and recovery it offers is worth emphasizing as it detect the node which has died by sending alert messages and redistributed the load of the dead node to all other nodes.

4. Facts and Dimensions

Facts are mainly referred to as the numeric values that represent a specific business aspect or activities, which are stored in Fact tables laying in centre of the star scheme a well-known database schema which contains facts that are linked with their dimensions [3], facts have an preference that they can be computed or derived at run time also they can be updated periodically with data from other operational databases.

The fact tables are huge tables which stores business measurements known as measures or facts [3]. They basically contain foreign keys to the dimension tables. For example if we have Customer Dimension and Sales Fact. Then the Sales fact will contain the Sales information of the various customers like profit, total sales etc.

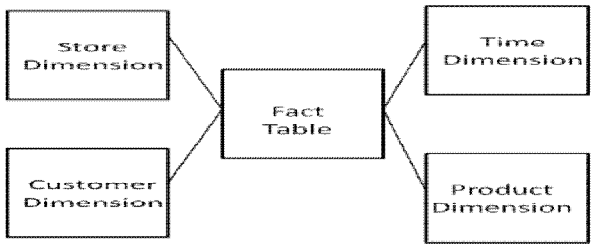

Figure 5: Star Schema.

Characteristics providing additional information of given fact data are normally stored in dimension table's where dimension attributes help to describe dimension value. They are explanatory in nature and transcript values are a part. Several distinct dimensions when combining several distinct dimensions with facts enable the user to answer a business question [3]. Every Dimension table has a Unique Identifier for each record which becomes the primary key for the Dimension tables.

Facts and Dimensions are the database objects and are most important components of Database schemas which helps in replicating data so that the system could become fault tolerant

Do not add page numbers.

5. Coherence Cluster

As we already studied about facts and dimensions we can conclude that facts are Big and Dimensions are small and we can say that facts would take more memory space than dimensions hence when we are designing any database application say in RDBMS then we have different fact and dimension table that would be connected as per the Fig 5 and for this purpose **SQL would be requiring lot of joins and key references** which would ultimately increase the overheads, consume lot of time and eventually lowers the overall performance of the application designed.

While when is to be stored in Coherence it is NoSQL that means SQL is not present therefore all the joins, relationships and ACID protocols are also missing and data is stored in some other fashion instead of tables nodes forming a cluster is present as shown in the middle layer of Fig 4, these nodes stores all the fact data across themselves and as said facts are updated periodically at run time from operational databases and the smaller data chunks are made dimensions which are replicated inside every node, as these dimensions are small thus use less space but due to this kind of replication the speed of the system rises tremendously, plus as there are nodes present in the cluster where each node could be a different machine as data is being stored in RAM of multiple machines hence at run time any number of machines could be added as a new node if the cluster this makes the system scalable also the backup of each node is saved by another as a backup which makes the system fault tolerant.

Thus Coherence had won over RDBMS over Speed, Scalability, availability and fault tolerance.

Take a look how data is stored inside a coherence cluster.

Thus from Fig 5 we conclude that we can divide any database application in two parts where one forms the facts and the second one for dimensions also facts are considered as the core from where the trade would start now as we know that fact tables are considered as the core tables then generally these tables accommodate large data as compared to that of dimension tables if we compare the amount of storage among the two kinds of table we would conclude that facts table are having comparatively large amount of data that the dimensions.

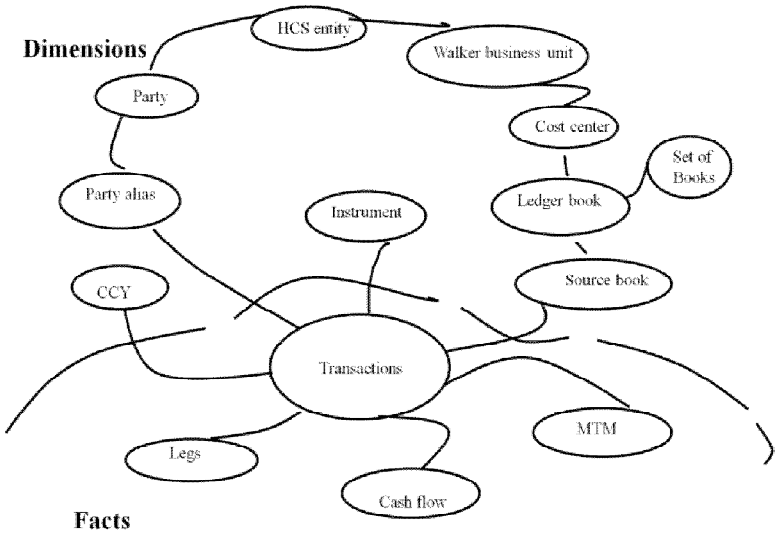

Figure 6: Facts and Dimension.

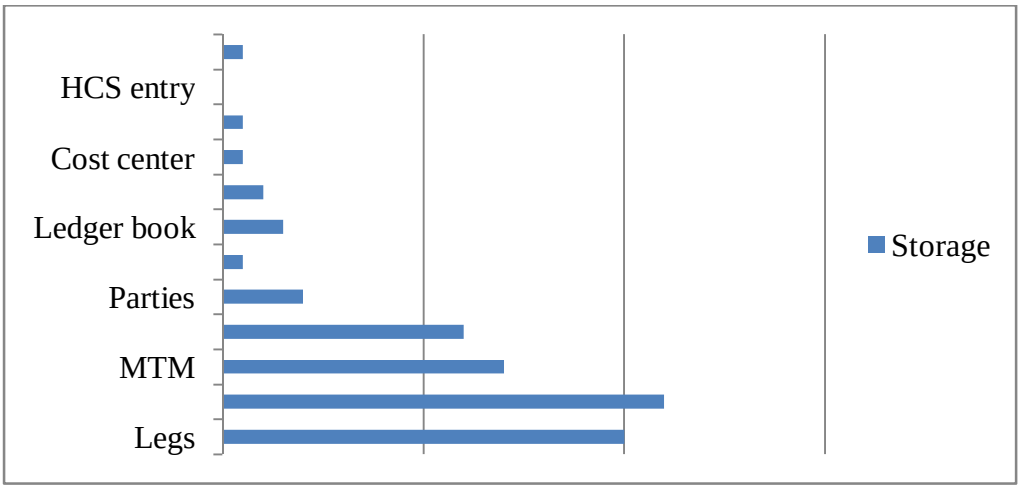

Figure7: Facts and dimensions storage.

It is clear that we are having larger amount of data stored in the facts table rather than the dimensions table as in fig 7, thus we need to focus on how we can scale up machines according to facts, we have to focus more on the facts that how we can make our system robust and fault tolerant as well as highly scalable by replicating the data set and which would require less among of storage space than those that would require higher storage space. Snowflake-Schema helps the designer to split the data into Dimensions (which are then replicated) and Facts (which are partitioned).

The first step is to represent the data model as a Snowflake-Schema. Take an object model such as that shown in Fig 6.The dotted line represents the division between Facts and Dimensions. It can be said that the Facts are the recorded fact, while dimensions represent the context that give that fact a defined meaning.

Dimensions are replicated here because they are small as compared to Facts and thus could be accommodated in each clustered node shown in fig 8 and Facts are partitioned into nodes so that they could be used for scaling up of the cluster.

Thus when we are designing any application in coherence we conclude that we have designed the design of the database according to our wish and requirement and when we design such kinds of databases then there is an ease while scaling up horizontal machines and increasing the storage space when such kind of applications are designed in both RDBMS and Coherence we conclude that the design of coherence is much more scalable, fault tolerant and gives better performance than the traditional RDBMS.

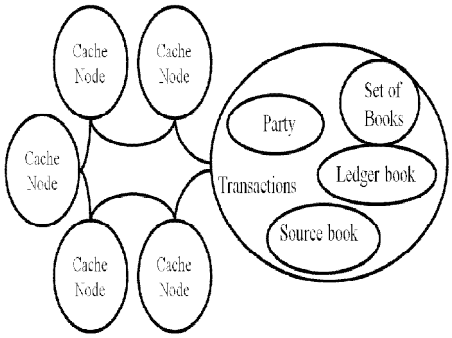

Figure 8: Clustered Nodes.

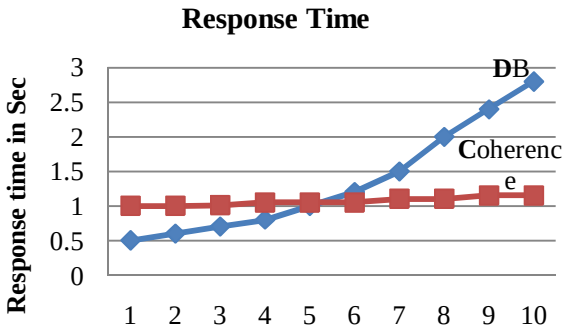

Figure 9: Performance Graph.

When both the relational database and Coherence are compared with respect to the response time they take while reading or writing any process of application then important results could be drawn as shown in the fig 9 performance graph.

6. Conclusions

RDBMS is limiting for many use cases because today's requirement is Speed and scalability. Therefore many newer databases are in use where applications are shifting towards In-Memory databases which could enhance the data access also people are moving towards such databases which could work according to our Use-Case.

NoSql is appropriate solution for those use cases where ACID is not considered that important and people are ready to stake ACID for speed and scalability.

There are various examples of NoSql databases like Cassandra which is the database prepared for the popular social networking site Facebook, Hbase (apache S/W foundation, modeled after Google's Big table), Mango DB (used for twitter) and many other Coherence is also one of them what have innovative features of its own (powered by Oracle, used by various multinational companies like RBS). Coherence has three important functions i.e. speed, scalability and fault tolerance. There are no single points of failure, and in case of any failure the load of a particular machine is distributed among others. Its redistribution is unbiased between the clusters, it works well for those use cases where partitioning of bigger data chunks and replication of smaller data chunks is required.

7. Acknowledgement

I extent thanks to Mrs Prakriti Trivedi for guiding me throughout my research work with her guidance teaching and suggestions motivated me for my work , I would also like to thank Mr.Chandervardhan Raghav, Mr Kuldeep Singh and Mr Bhairon rathore for their valuable support.

References

[1] Tudorica, Bogdan George Cristian and Bucur *A comparison between several NoSQL databases with comments and notes*, IEEE Conference on Commerce and Enterprise Computing. April 6-7,2011

[2] Jing Han, Haihong E, Guan Le and Jian Du, "Survey on NoSQL Database", *IEEE Conference on Cluster Computing.*2011.

[3] Michael stonebraker, Ugur Cetintemel, "One Size Fits All": An Idea Whose Time Has Come and Gone, 10th IEEE/ACIS International Conference on Computer and Information Science, 2011.

[4] P. Malik and A. Lakshman, *Cassandra-a decentralized structured storage system.* SIGOPS Operation System. Revolution., 44:35–40, April 2010.

[5] N.Leavitt, *Will NoSQL Databases Live up to their promise?* IEEE computer Society, vol. 43(2),2010,pp.12-14.

[6] J. Ernst,*SOL databases v. NoSQL databases,*Comm of ACM, vol.53(4),2010

[7] Jing Han, Meina Song and Junde Song, *A Novel Solution of Distributed Memory NoSQL Database for Cloud Computing*. 10th IEEE International Conference on Computer and Information Science. June 2011.

[8] Bucur, Cristian; Tudorica, Bogdan George, *Solutions for working with large data volumes in web applications*, The Proceedings of the IE 2011 Education, Research and Business Technologies, International Conference, 5-7 May 2011.

[9] Understanding distributed and in-Memory architectures. http:www.benstopford.com/2011-/08/14/distributed-storage-phase-change-memory-and-the-rebirth-of-the-in-memory-databases

[10] Cook, John D., "ACID versus BASE for database transactions", www.johndcook.com/blog/2009/07/06/brewer-cap-theorem-base/.