International Journal of Computer Science and Software Engineering (IJCSSE), Volume 7, Issue 8, August 2018
ISSN (Online): 2409-4285 www.IJCSSE.org Page: 179-185

Application of Improved Intelligent Load Balancing Model in a Replicated Database Environment

Ekong A. P.[1], Nwachukwu E.O.[2], Onyejegbu L. N.[3]

[1, 2, 3] Department of Computer Science, University of Port Harcourt, Rivers State, Nigeria

[1]anietiekong@yahoo.com, [2]enoch.nwachukwu@uniport.edu.ng, [3]laeticia.onyejegbu@uniport.edu.ng

ABSTRACT

When request are randomly assigned to database replicas in a replicated database environment, it causes a situation where in some databases are heavily overtasked while others sits idle. Intelligent Load balancing is highly essential in a replicated distributed database architecture in achieving a higher throughput making sure that no single database is over-utilized, and by so doing improving the overall throughput of the system. Several techniques are proposed or deployed for load balancing but most of them makes the ideal demand for equity or fairness to still be farfetched and are plagued with several constraints such as fewness of parameters in consideration, the failure to reassigned executing processes on failed servers to active and suitable ones and the non-consideration of issues relating to replicated databases all which lead to an increase in the time spent by processes in the system. In this article, An Improved Intelligent Load Balancing (ILB) Model for a replicated database Environment has been proposed. The aim of ILB is to provide fairness to all the Databases by balancing the request among the database replicas. In our proposed solution, the number of requests attended to at any instance on the individual database in the cluster and data location is also taken into consideration and their impact on total time spent by a queued request in the system is determined. We have developed an improved intelligent load balancing algorithm which balances the load or requests on different databases in a homogenous database cluster. The result obtained showed that in replicated database, there is more 70% less time spent to attend to users request using Improved Intelligent Load Balancing than using other existing algorithms hence solving the problem of load imbalance and unfairness in request distribution.

Keywords: *Improved Intelligent, Load Balancing Model, Replicated, Database Environment.*

1. INTRODUCTION

In today's world, there is a heavy dependence on data and information, everyone depends on one database or the other for information. Most business cannot function for few seconds without these information. Even our schools and hospitals are not left out. Recent virus by ransomware attacks in many countries in the world have revealed the degree of dependence on data by the society, several hospitals' booking had to be canceled as a result of the locking of patients' record in the computer by the ransomware. While it remains a possibility for one to have all of databases concentrated in one computer with universal access by authorized users via network both local area and wide area including the Internet and although the management of such a centralized databases can easily be done with much benefits, it poses a lot of problems as well. For instance, if the central server goes down, the clients are disconnected from the databases. Also the communications costs from the numerous clients to the central server can be very expensive. One solution to such problems, and an alternative design to the centralized database concept, is known as replicated database. The idea is that instead of having one, centralized database, the database is replicated into many other locations in the network, each of which has its own computer and data storage facilities. All of these replica data are similar to each other in all aspects all and all function as a single logical database. When the database is queried, the location from which the data comes from is not necessary known to the user that is the database location is transparent to the user. Database replication is therefore the copying of data from a database in one computer or server to a database in another so that all users share the same level of information. The database units know as replica can function independently and cooperate with each other to provide services to attend to request by clients from all locations.

Load balancing being the distribution of workloads across multiple computing resources, such as computers, a computer cluster, network links, central processing units or disk drives easily comes to focus here. In a replicated database environment, load balancing is the distribution of request load across multiple database replicas. Load balancing in a database management system main goal is to optimize database use, maximize throughput, minimize response time, and avoid overload of any single database server or reduce calls to any single database location. Improved Intelligent load balancing in replicated database management is the automatic selection of a database based on least cost consideration. This cost includes but not limited to the host server configuration and status and the state of the database engine at the instance of the request.

2. LITERATURE REVIEW

Load Balancing being the distribution of workload among different computing resources in order to achieve optimal utilization of resources hence increasing system throughput and minimizing total system response time is used to avoid overload on the resources and for sharing traffic among different servers *Nusrat et al. (2014*). Using Load balancing, Data communication can take place with minimum delay. It is used to minimize waiting time. In the clouds, load balancing is used for balancing load on virtual cloud machines and other resources resident there (*Nusrat et al., 2014*).Intelligent load balancing is the automatic selection of a system based on the server configuration and status and assigning the job on queue to the most appropriate server.

3. LOAD BALANCING AND REPLICATED DATABASES

A replicated database is a system comprising of many and similar copies of a database distributed across the same or different locations. Each copy of the database is a replica of the original one and can work independently to attend to user's request. The database can 'liaise;' with each other to enable the whole system perform the intended functions. Clients can send any type of operations or query to the system, and it is left for a coordinating agent to ensure that the request are fairly distributed among the various database replicas.

The major concern of database replication is consistency which is how to make sure the data among the replicas are consistent or similar after changes are made in any of the copies. The solution to this is that if there is any update in any of the replicas, such updates must be propagated across all other copies. Moreso, there must be adequate and coordinated mechanism in place to ensure that no two records are simultaneous written to. If two operations must be performed on a record, one should be a read while the other a write operations. This can be implemented by the use of appropriate semaphores to identify a record with ongoing write operation. A middleware based approach can be used to perform the dual operation of necessary coordination for concurrency control and Load balancing in a replicated database environment. The middleware sits between client and replicas of the database as illustrated in figure 1.

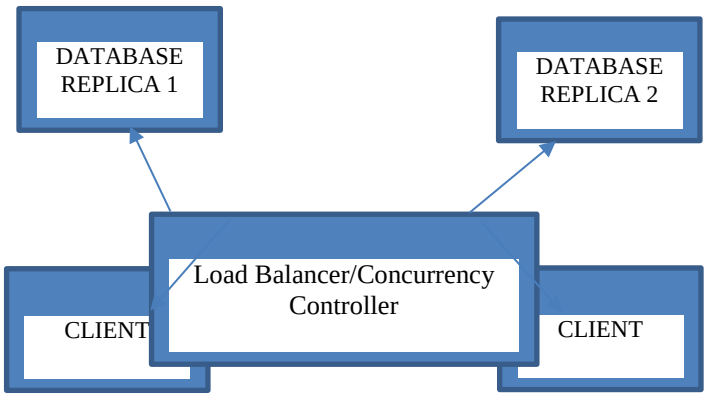

Fig. 1. Intelligent Balancing/Concurrency Control in Intelligent Balancing in Replicated Database Environment

Load Balancing in a replicated database environment is a relatively new technique that ensures maximum throughput while at the same time minimizing response time. When traffic or tasks are divided between databases, data can be sent and received and tasks can be completed without major delay.

Different kinds of algorithms are available that helps traffic directions to available servers, be it database servers or database engines or any other server. A basic example of load balancing in our daily life can be related to internet. Without load balancing, users could experience delays, timeouts and possible long system responses. Load balancing solutions usually apply redundant servers which help a better distribution of the communication traffic so that the website availability is conclusively settled [*Amanpreet C. et al. (2014)]*.

To manage the jobs in a multidatabase systems the selection mechanism has to resolve to avoid a typical control conflict. The decision process in which the actions are determined by where to transfer the just requested query to has to be resolved. Typical solution for load volume of each database copy of the setup is weighted by the same amount of the request over long time. This conflict can be controlled by reinforced learning.

Traditional load balancing techniques are not much efficient as to provide healthy environment as per today's requirement. According to (Hendra R. et al., 2009) most of the loads balancing techniques used these days are static and hence not suitable according to today's environment. By survey the pro and cons of different load balancing algorithms, dynamic load balancing algorithms can manage the load on different servers.

Load balancing in a replicated database environment**,** is the distribution of requests across multiple database engines resident in a single machine or on multiple machines in a network and can only be achieved using proper scheduling algorithm.

4. SCHEDULING

Scheduling is organization, choosing and timing all resources and their usages to carry out comprehensively the activities needed to produce the required outputs at the desired times, while at the same time satisfying a several time and interaction constraints among the entities and their activities.

For efficient load balancing, the issues of scheduling must be handle in a systematic way. The scheduling discipline such as First In First Out (FIFO) where requests or processes are attended to in the order which they arrived, Earliest Deadline First (EDF) where request or process that has nearest deadline will first be attended to, Shortest Remaining Time First (SRTF) where the request or process with the lowest remaining execution time is first attended to, Pre-Emptive Scheduling (PES) where a request or process is attended to, based on a weight or priority, arbitrarily assigned to such request or process Round-Robin (RR) where the requests are being attended to using a fixed time slice basis in a cyclic fashion must be effectively used.

Whatever process or request 'elected' to be the next to be served, an appropriate algorithm will have to play the role of a coordinator or balancer and will decide the databases it is going to assign the 'elected' request or process to, based on the status of the database at that particular instance.

5. APPROACHES TO INTELLIGENT LOAD BALANCING SCHEDULING

There are different approaches to intelligent load balancing, most of the approaches make use of single metrics in making decisions in a server clusters. After critically studying these different approaches, we were able to identify several issues which inspired us to develop an improved version. The existing approaches are as discussed in ensuing paragraphs.

5.1 Ram Based Approach on Load Balancing

(Amritpal S. et al., 2014) discussed an algorithm for load balancing that is dynamic. He noted that in the cloud, VMs are of varying configuration with respect to the user requirement and ordered based on Random Access Memory (RAM). When the balancer receives a request, the request is given to the machine with the highest available RAM. The balancer also has a database of the priorities and the request or process count currently running or allocated to the each virtual machine.

He concluded that good load balancing provide performance benefits and that by introducing a new model for load balancing server conditions can easily and dynamically be adapted to hence solving most of the traffic related issues. By testing the model using the metrics of time of allocation, and responsiveness, the Load balancing Model works very efficiency.

5.2 Load Balancing In the Cloud (DODDINI P., 2013)

(Doddini P., 2013) discussed the requirements for cloud computing with regards to factors such as access control, migration, security, etc and also reviewed the models employed in the clouds for balancing loads; such as weighted active monitoring, dynamic and static load balancing algorithms, ant colony optimization models, and distributed systems load balancing. He attempted to identify the problems in the existing models using some metrics and noticed that they were not performing

optimally as a result of improper implementation of load balancing models

5.3 Load Balancing In the Cloud Data Centers By *Hemont Et Al.,* (2013)

(Hemont et al., 2013) researched on Load Balancing on Cloud Data Centres. He analysed the load balancing models and their respective implications using a cloud simulating tool, cloudsim. In his research, different user communicated with the data centres and the output was produced. He calculated the maximum and minimum time together with data migration cost. He concluded in Equally Spread, Round Robin and Throttled Load balancing, the request time were similar. Moreso, calculated cost for Virtual Machine usage is same for Equally Spread and Round Robin but Throttled Load balancing algorithm showed a reduction in the usage cost implying that the Throttled Load balancing algorithm works more efficiently in the cost metrics for cloud load balancing.

5.4 Cpu Based Load Balancing Algorithm By Nayandeep Sran

(Sran et al., 2013) proposed a non-dynamic scheduling model with overhead due VM selection and considered only one factor. He further discussed the two types of policies for VMs: consolidation and migration. He gave priority to Virtual Machine migration policy using resources at its disposal. At first, the CPU utilization of the machine is checked 80% here is an arbitrary value chosen the threshold for the machine being overloaded. If utilization of the machine is greater than or equal to 80%, that machine was considered to be over utilized and underutilized if else.

Finally, the algorithm allocates priority to virtual machines and the migrated to such machines are thereafter done according to the highest-priority- first method. He noticed from analysis that there is a reduction of the average response time. He concluded that the proposed algorithm can balance the load much more than the existing algorithms. His Load Balancing algorithm was targeted at providing dynamic balance of resources on demand to accomplish any required task. This algorithm also guarantees better security, by focusing on hiding personal details of the cloud users from the cloud provider known as identity based security.

5.5 Round Robin Load Balancing Algorithm (Nusrat Pasha Et Al., 2014)

Nusrat analysed Round Robin approach to Load Balancing Algorithm in the Clouds, He proposed an improved version of the algorithm. In this, he assigned the minimum time, maximum time to different Virtual Machines (VM). From there, he noted that the RR performed when overall response is considered and the request small and moreover with number of Virtual Machine set and otherwise when there is a large VM set. Also, by the application of service broker policy, the issue of deadlock and server overflow was greatly minimized. He concluded that the proposed algorithm shows that there is improving in the overall performance with respect to time consumption in scheduling.

5.6 Limitations Of The Above Algorithms

We have observed that the above algorithms has one limitations or the other. The round robin algorithm is a static algorithm and hence highly limited in its applicability. While the other algorithms were based on hardware parameters; the CPU and the RAM. Also, the algorithms did not answer questions or proffer solutions for node failures during process executive ie where a server 'dies in action'. Moreso, the algorithms never handle balancing that involved databases and hence could not be assumed to effectively proffer solutions to balancing problems associated with databases and particularly replicated or distributed databases. Furthermore, the idea that the load could be balance even when one server was not in focus as the major concentration was only on multi server environment whereas load could be balance within a server with consideration to any multifaceted parameter, be it CPU or database. Finally, the algorithms does not handle certain cases like the threshold, a value above which the highest priority node is selected, is kept constant at 80% which may not always be the case, as network administrators policies on when a server could be considered busy could vary based on different expected performance.

6. PROPOSED SYSTEM

Our effective goal is to design an improved intelligent load balancing system for a replicated database environment that has significant improvement over a non intelligent or static approach.

For many replicated database environments, deploying software to manage scheduling is mostly and traffic is mostly not considered by IT center by administrators.

They tend consider performances only equipment's hardware specifications—port cost, average failure rates, replacement and whatever does not fall under these yardstick is useless.

We propose an intelligent model which will introduce intelligence into balancing in a replicated database environment: dynamically learning the database state and configuration and sending the process to the less busy database engine, which will offer a remarkable improvement over the previous or existing models that used an entirely static approach no balancing at all. Figure 2 is an illustration of the proposed model.

Here, the necessary input are request or processes and is sent to the appropriate database engine by the balancer based on the learnt database status. At the end, the total time spent by processes in the system is computed and placed side by side with the time spent using the existing load balancing models. Based on this, appropriate decisions on improving the service delivery like increasing the number of replicas can be made. For example, it will be clear if a request waits longer than necessary in the queue based on the arrival time and the leaving time

6.1 Mathematical Model of the Proposed Improved Intelligent Load Balancing Model

Assume replicated databases linked with a network or resident in a single system. Consider the following description of the set of dynamics $N=\{R_i,B,D_i\}$, with R_i representing the various request on database by clients processes, B the Balancer and D_i representing the different databases. Arrivals of requests P_i to B follows a poisons distribution and are assigned to most suitable D. R_i are of different types and arrivals are independent. Here, we assume that B has infinite capacity implying that B can accommodate every R without delay.

In the model it is also assumed that requests do not change their types until they depart from the system but request, p, can be assigned to another database $D \in N$ if the executing database is down.

We have to repeatedly monitor the status of the database replica to know if it is busy or not since it is on the basis of the status that we are transferring the request to balance the load. In finding the Status of a database system we require the number of running request.

First,
Input: T (threshold)
Expected Output: Utilization of parameter
Let the database be defined as set: $S = \{Z_1, Z_2, ..Z_n\}$ and their respective utilization value as set $V = \{V_1, V_2, ..V_n\}$ and their threshold (T) as set $T = \{T_1, T_2, ..T_n\}$, where for each S_n, T_n, $n \in [0,100]$ and V_n being a Boolean of either underutilized or over utilized
Next,
Input: 1. Database utilization threshold, T_n
Output: Database Status Information as ;
Busy: If $V_i >= T_i$
Not Busy: If $V_i < T_i$

Compute the delay (D) as (D_s-D_p)
where T_p is Time request is made and T_s is Time process is attended to
Compute the cumulative time used to attend to all process or to assigned to the suitable databases

$$CT = \sum_{i=1}^{n} Di$$

Where CT is the total time used by the balancer to assign a request to the most suitable database engine.

Note: The replica is not available and cannot receive jobs until it returns to the Non Busy state. The balancers create the status table and the load count are input into the Load Status Tables. The balancers use the table to calculate the database status. When a request arrives at the balancer, the request is assigned by balancer to the database according to its current summative utilization value. As the values changes based on the number of assigned request, the status of the database as noted by the balancer also changes.

6.2 Algorithm of the Proposed System

```
Load balancer_on_best server ()
{
Initialize all the Dbase status to LESS BUSY in the DBASE status list;
Initialize balancer database with no entries;
While(new request are received by the balancer)
Do {  queue the requests;
    If( DBASE  status == AVAILABLE)    {
        If (DBASE utilization>=threshold in percent )
                { Designate DBASE as BUSY DBASE }
                Else
                Designate DBASE to LESS BUSY DBASE
                Check require database requirement
                if (request requirement<=DBASE specification && request priority=highest)
```

```
                { send process request to first less
        busy dbase }
    }
        Remove request from request queue
        Update balancers request table with executing
        request
        Start request timer
        If execution time>maximum timer
        {
        Return dbase engine or server dead
        Reassign request to next dbase meeting the
        requirement
        Update relevant databases
        }
        update the balancer request table with request
        complete status
        Reinitial server status && return database
        available
                }
    Else
    {       Leave the request on queue;       }
  } While request queue<>empty
}
```

6.3 Result

Table 1: Cumulative Response Time for Round Robin and the Intelligent Algorithm

Database name	Static_Response_Time	Improved Intelligent Response_Time
Dbreplica1	**20s**	**5s**
Dbrelica2	**15s**	**7s**
Bbreplica3	**15s**	**10s**

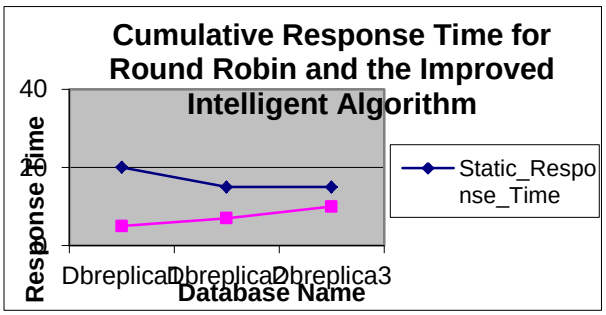

Fig. 2. Chart Illustrating Cumulative Response Time for Round Robin and the Intelligent Algorithm

We have demonstrated a load balancing model that shows an improved intelligence over algorithms with respect to load balancing and which considered some factors in selecting a database in order to achieve good service rate to waiting request. I worked on request migration policy by giving priority to databases on the basis of their status which is affected by the number of request it is currently attending to and the resources available to it at any given time. Firstly, the algorithm checked whether the executing requests is equal to, greater than or less than a certain set threshold and which can be adjusted based on the administrators discretion. If the database utilization is greater than or equal to the threshold then that database engine was understood to be busy or over utilized and no further process(s) will be allocated at that instance. But if the utilization is less than the threshold then that database is noted to be underutilized by the balancer and the next request in queue is allocated.

In the next step of the algorithm. If the server does not respond over a period of time then the database server is declared unavailable and the assigned request can be allocated to another database. We noticed that from analysis of the Load Balancing Algorithms, we were able to reduce the total time used by a database engine clusters to attend to all request and were able to intelligently balance workload across the databases in a replicated database scenario as the flow of work is now sequenced based on database status which is calculated based on the metric, number of executing request on each database replica, as 'understood' by Load Balancer.

We established a prototype system to simulate the replicated database load balancing system. This system consisted of three (3) homogenous virtual databases (database replica) and five(5) clients requesting system. In our simulation, we abstracted the physical network of computing. We supposed that all the database are homogenous and their initial working loads are at same level. When the requests on a database overweighs its capacity as set in the threshold, the database is noted by the balancer as busy.

In the experiment we submit requests from the virtual clients to the request queue in the balancer. At this time the database status based on the number of requests it is already handling has been learnt by the balancer and designated as busy or not busy depending on whether the matric under consideration is greater that the set threshold and as being updated frequently by the balancer. Based on this status, the balancer migrates the requests from the submitted queue to the appropriate database and add it to executing request queue.

As a control, at the same time the database mirror is also assessed by the balancer based round robin algorithm fashion, which is a static approach to load balancing. We had to test the algorithms simultaneously as a control experiment must be tested with the same conditions as the main experiment and since the variables are continuously changing, the algorithms has to be tested at the same environmental instance. The generated values

in table 1 are being upload into a database and the results shown as graphs in figure 1.
Whereas an average of 7.3 seconds using Improved Intelligent Load Balancing Algorithm, 16.6 seconds was spent using the static load balancing alternative, this clearly shows a great reduction in the time spent by requests in the system even when the request load is uneven due to the intelligence applied in allocating such request based on the database usage status.

7. CONTRIBUTION TO KNOWLEDGE

This research has contributed to knowledge in the following ways:

1. An improved model for intelligent load balancing System for a replicated database environment has been developed
2. it has solved the problem of process dying in action by ensuring process reassignment to other servers
3. Requests are assigned appropriately instead of blind request assignment as in a non-intelligent scenario.

8. CONCLUSION

In this paper, an intelligent load balancing model for a replicated database environment was looked at. The implication of static balancing model, their weakness and strength are comprehensively x-rayed and this paper has showcased how they fare unfavourably when placed side by side with this improved and intelligent load balancing model. The primary objectives of load balancing which is to optimize server use, maximize throughput, minimize response time, and avoid overload of any single resource have been comprehensively met through the use of multiple homogenous or replicated databases with load balancing instead of a single database. This has also led to increase in reliability through redundancy. Based on the results, it could rightly be inferred that the improved intelligent balancing algorithm is better in the way in which the requests are being allocated to the databases and the time it takes for the request queue to be exhausted that its static alternative.

REFERENCES

[1] Amanpreet C. & Navtej S. (2014). A review study on cloud computing and Load Balancing algorithms, International Journal of Advanced Research in Computer Science and Software Engineering, 4(4).
[2] Amritpal S. (2014). Comparative analysis of proposed algorithm with existing load balancing scheduling algorithms in Cloud Computing , International Journal of Advanced Research in Computer Science and Software Engineering: (3)(1). ISSN 2278-6856
[3] Doddini P. (2013). Load Balancing Algorithms in Cloud Computing. International Journal of Advanced Computer and Mathematical Sciences. ISSN 2230-9624. 4(3), 229-233.
[4] Hendra R., Yudi S. (2009). The Simulation of Static Load Balancing Algorithms, International Conference on Electrical Engineering and Informatics, 640-645, 5-7.
[5] Nusrat P., Amit A. & Ravi R. (2014), Round Robin approach to VM load balancing algorithm in clouds computing environment. International Journal of Advanced Research in Computer Science and Software Engineering 4(5),34-39
[6] Hemont S., Parag R. & Vinay C. (2013) Load Balancing On Cloud Data Centres International Journal of Advanced Research in Computer Science and Software Engineering (1),January - 2013, 1-4, 3(1), ISSN: 2277 128X
[7] Sran N., Navdeep K. (2013) Zero Proof Authentication and Efficient Load Balancing Algorithm for Dynamic Cloud Environment International Journal of Advanced Research in Computer Science and Software Engineering 3(7):1327-1332.