Middleware-based Database Server Allocation of Distributed Database on PC Cluster Systems

Junar A. Landicho* and Consorcio S. Namoco, Jr.
College of Industrial and Information Technology
Mindanao University of Science and Technology
Cagayan de Oro City, 9000 Philippines
**junar_landicho@yahoo.com*

Date received: June 01, 2013
Revision accepted: January 15, 2014

Abstract

The study seeks to provide a distributed database using the Middleware-based Database Server Allocation (MDSA). The MDSA allocates available servers to cluster computers and search for the best cluster that caters to a query. There were eight experiments performed using MDSA and were compared to the sequential and random search method, which include controlled CPU utilization and random CPU utilization in terms of access time, single query and multiple queries. Experimental results showed that with MDSA, there is a reduction of data response time under varying number of nodes, ranging from 1 to 8 clustered servers.

***Keywords**:* database, distributed database, middleware, PC cluster

1. Introduction

A distributed database is a collection of databases that can be stored at different computer network sites. Distributed Database Management System (DDBMS), on the other hand, is a software that manages a distributed database while providing an access mechanism that makes the distribution transparent to the users (Silberschatz *et al*, 2006). Figure 1 shows the sample of distributed data and distributed processing.

In Personal Computer (PC) clustering system, the distributed data environment approach is used where the information will be decomposed into smaller pieces and then distributes them to be stored on several data storage nodes in the cluster computer systems. Those cluster computers are connected together as a high performance computer. The DDBMSs will handle those pieces of data as one unit. Users do not need to know where exactly the data will be stored on cluster nodes (Amiri, 2004; Johnson and Anthes, 2003; Brown *et al.*, 2007).

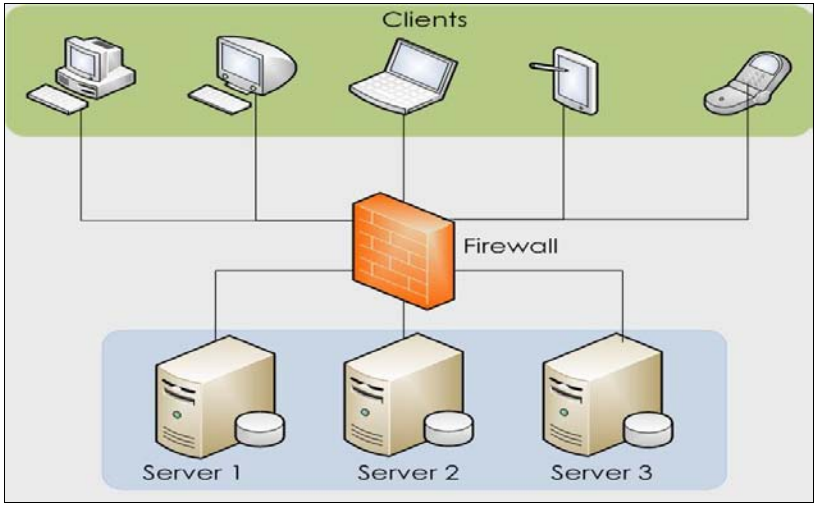

Figure 1. Distributed data and distributed processing

Figure 2 shows the sample of design in PC cluster system in distributed data approach. There are several advantages of this approach which include higher processing performance, more memory capacity, higher network bandwidth and higher I/O bandwidth. There are also several storage nodes; each storage node will handle their own pieces of data in parallel. Furthermore, each of them has its own network interface card that handles network bandwidth. The best case of network bandwidth can improve the system by the product of the number of storage nodes and bandwidth of each network interface.

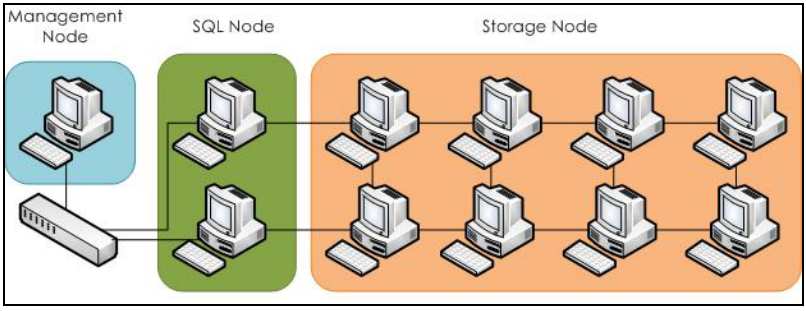

Figure 2. PC cluster system in distributed data approach

In this approach, it is more efficient to use a middleware that can allocate the

available servers among the clustered computers, and offers the best cluster to serve for a query. A middleware is a software that mediates between an application program and a network. It manages the interaction between applications across heterogeneous computing nodes. Using middleware for distributed database improves the query processes, increases the portability of the queries, and improves the system maintenance and reliability. The middleware can also provide fault-tolerant control among the clustered servers (Pukdesree *et al.,* 2010; Oracle Corp., 2010)

This study is intended to design an efficient tool to help achieve an effective system through the distributed database on PC clustered systems.

Figure 3 shows the middleware-based database server allocation diagram. First, the middleware-based database server allocation accepts connections from the clients. All the connections are done using Transmission Control Protocol / Internet Protocol (TCP/IP). It is one of the protocols of the Internet Protocol suite.

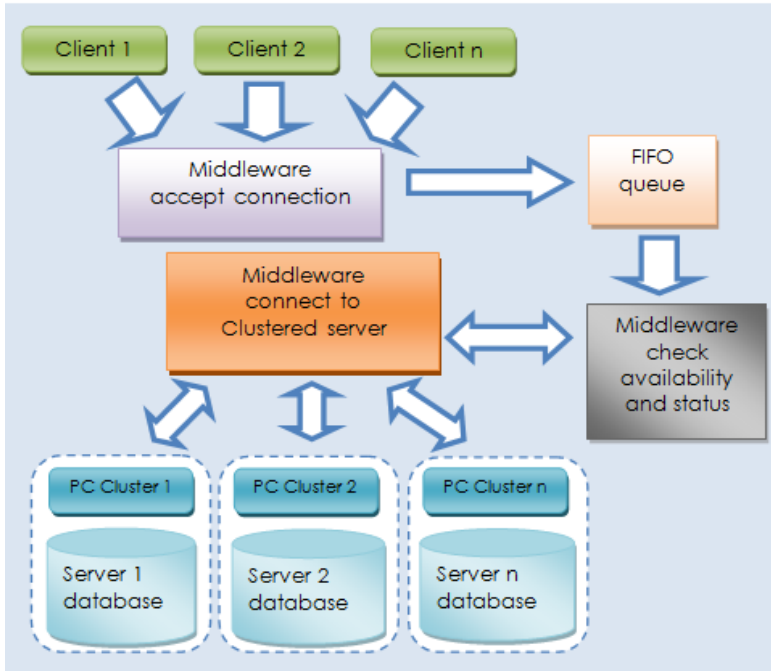

Figure 3. Middleware-based database server allocation diagram

Using TCP/IP, applications on network hosts can create connections to one another, over which they can exchange data. The protocol guarantees reliable and order delivery of sender to receiver data. The connections are done in a First in First out (FIFO) manner. The first client to connect to the middleware is the one who will connect first to the clustered server. Next, the middleware will choose a clustered server through an IP address. Every server has an IP address which is unique. The client has a unique IP address in order for the server to determine the source address. After choosing a clustered server, the middleware tries to connect to it. If the server will respond, it means that it is available. The middleware will then get the Central Processing Unit (CPU) utilization of each clustered server (CS). The CPU utilization is the basis for determining whether the CS is busy or not. The CS that has the highest CPU utilization is considered as busy while the CS that has lowest CPU utilization means less query process. When the middleware can find the CS that has lowest CPU utilization, the middleware will forward the IP address of the CS to the client. The client will now establish a connection to the CS using the TCP/IP and MySQL Cluster. Then the client sends the query to the CS. The CS processes the request and sends back to the client. A process is an executing program that performs some useful operation of data in a computer. After processing and sending back the request to the client, the middleware will close the connection of the client and CS. Every time the client requests to the clustered servers, the middleware will find first among the clustered servers one that has lowest CPU utilization.

2. Methodology

2.1 Network Model

The network model used for MDSA is shown in Figure 4. The model includes connectivity (topology), bandwidth and number of PC used. Each CS was limited to three PC for the purpose of evaluation and minimum requirements in creating MySQL Cluster Architecture.

2.2 Database Cluster Model

In this study, each MySQL Cluster database was split over 3 Ubuntu machines having these characteristics: (a) 2 hosts with each running 1 data

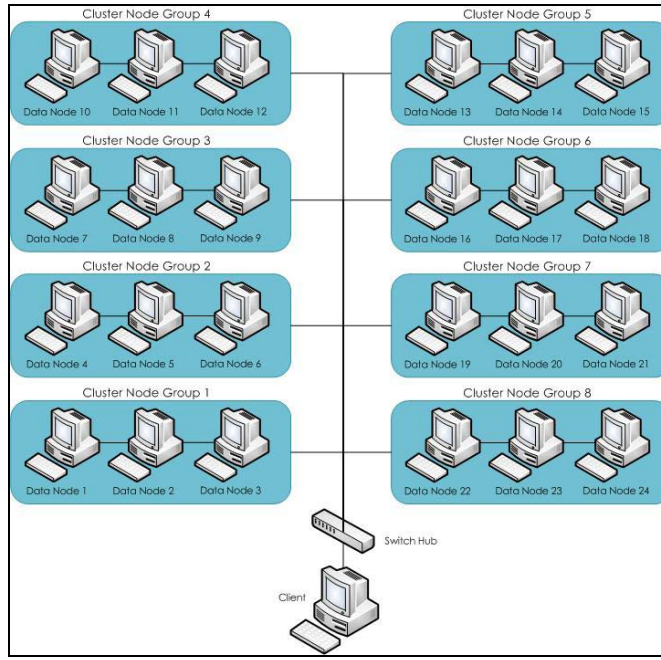

Figure 4. Network model of MDSA

node; (b) 3rd host running management node and; (c) 3 hosts running MySQL Servers. This is shown in Figure 5.

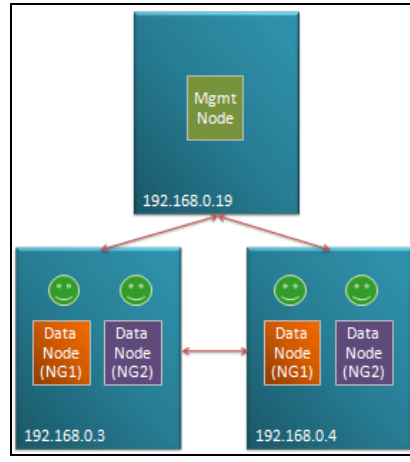

Figure 5. Database cluster architecture model

2.3 Hardware Requirements

The MDSA proposed in this study used twenty-four computer workstations. The specifications of each computer are a single Intel Atom CPU D410, 2 GB MHz of RAM, 200 GB-SATA II of hard drive and on boarded 100 Mbps of network interface. All computers were connected using one 100 Mbps twenty-four-port switching hubs. The system is a closed system that prevents other factors from affecting the results of the experimental. Although the computers used in setting up the system did not have high-end specifications, it did not affect the functionality of the system. In the distributed data approach, the network bandwidth is a very important factor that will affect the results of the experiment. The standard bandwidth speed for most PC was 100 Mbps and UTP cable specifically CAT5e that has maximum 350 Mbps of network capacity. The system proposed in this study used CAT5 and CAT6 or CAT7 that has 550 Mbps.

2.4 Software Requirements

Ubuntu Linux 10.0 was used for the operating system, since it is the most reliable, potential, stable and secure operating system (OS). The proposed system has not been tested on other open source OS such as fedora or FreeBSD. Ubuntu 10.0 also provides support for their customers via subscription patches or packages that can also update via internet. Hence, the system administrator can fix or upgrade the system software. In this study, MySQL Cluster 7.0 was used as distributed DBMS which is the latest version at that time. MySQL Cluster 7.0 provides many advance features as enterprise DBMSs such as HA or online duplication of database. The required packages were only installed on each type of MySQL Cluster components, for example management node, SQL node and storage nodes. MySQL Cluster supports both disk-based and in-memory database. This study used in-memory database approach that performs better and faster compared to disk-based approach. MySQL Cluster 7.0 also supports up to eight threads in parallel, that is very suitable for present processor's multi-thread or multi-core era.

2.5 Algorithm for MDSA

Flowcharts are often used to represent algorithms as shown in Figure 6. This algorithm is meant to connect the nodes in the available server through MDSA.

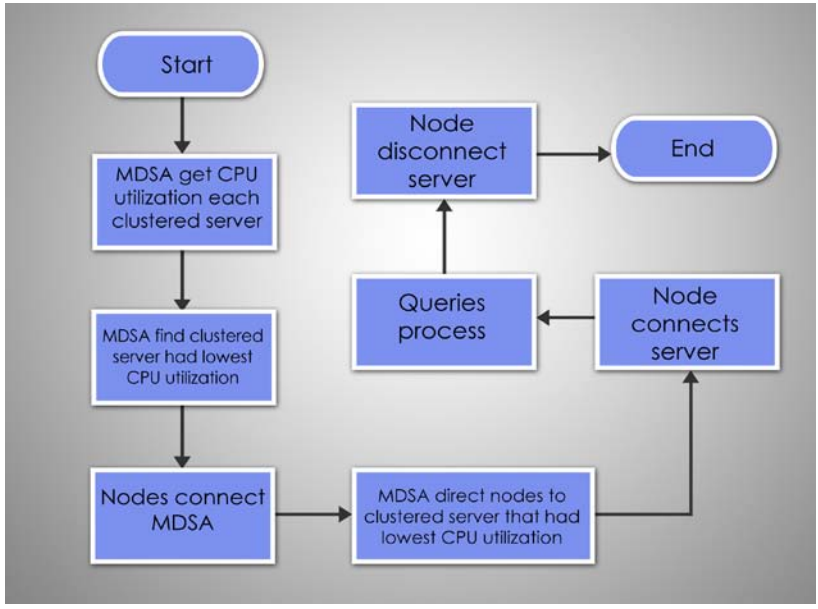

Figure 6. A flowchart showing node connects to server using MDSA

2.6 Search Methods

Three search methods were used to determine which of the CSs would receive any given request. In the sequential method, the requests followed a set sequence such as CS one, then CS two, then CS three, then CS four, until it reached to the last CS then back to one. The random method used a random number generator to select a CS randomly from the pool of CSs. It is hoped that if the number generator was truly random, the work load would get evenly distributed. The MDSA method monitors the OS on each potential CS node to ascertain its current load in real time. CSs under heavy loads, which were unable to report in a timely interval, were assumed to be at 100% utilization and busy. Selection was based on the lowest utilization currently reported.

2.7 Performance Evaluation

Effective evaluation of MDSA uses different scenarios. There are two major setups for the evaluation, CSs during controlled CPU utilization and CSs during random CPU utilization. Both CPU utilizations used three search

methods, such as sequential method, random method and MDSA. In controlled CPU utilization, a specific CS determines which has lowest CPU utilization. In random CPU utilization, each method does not have the same CS which has lowest CPU utilization.

The methods mentioned above were evaluated through response time in accessing the CSs, response time in assigning single query to the specific CS and response time in assigning multiple queries to the specific CS. The system was tested and evaluated in a single run having twenty-five PC, where twenty-four for cluster nodes and one for a client.

3. Results and Discussion

3.1 The MDSA

Screenshots were taken after determining the response time in different sample test. Figure 7 shows the CPU Utilization of each CS. It displays which CS has the lowest CPU utilization and is connected to the Clustered Server.

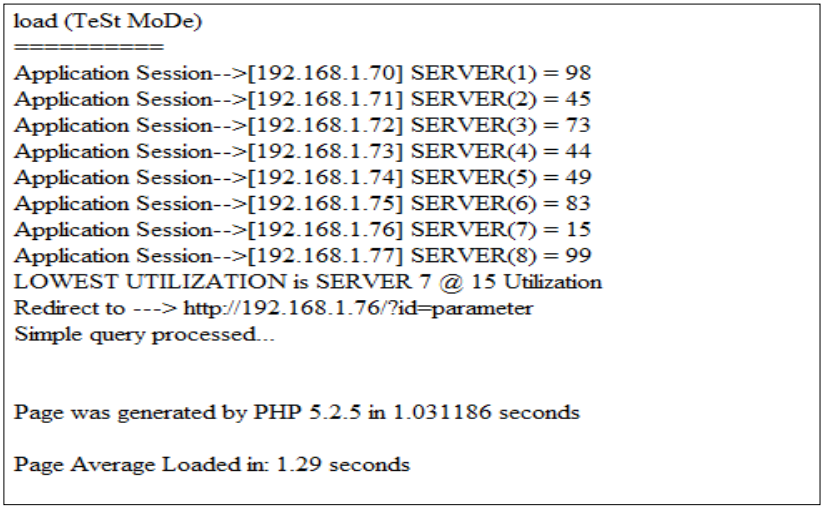

Figure 7. A screenshot showing the lowest CPU utilization in PC cluster

3.2 Percentage Time Difference on MDSA as Compared to Sequential and Random Search Method Controlled CPU Utilization

Table 1 and 2 show the percentage time difference on MDSA as compared to sequential and random search method during controlled CPU utilization and random CPU utilization, respectively. In general, the MDSA appears more efficient compared to sequential method and random method in terms of access time, single query and multiple queries.

Table 1. Percentage time difference on MDSA as compared to sequential and random search method during controlled CPU utilization

No. of Servers	Access Time		Single Query		Multiple Queries	
	With Respect to Sequential Method	With Respect to Random Method	With Respect to Sequential Method	With Respect to Random Method	With Respect to Sequential Method	With Respect to Random Method
1	-1.692	-2.098	32.034	2.417	18.116	31.878
2	49.252	-1.597	50.153	8.805	53.745	5.878
3	66.189	49.351	67.440	53.819	45.878	31.090
4	74.994	50.100	74.920	29.661	67.309	34.879
5	79.976	49.950	78.299	77.984	80.328	86.894
6	83.056	74.544	77.409	66.618	80.931	79.147
7	85.576	-0.997	80.588	72.126	84.309	2.052
8	87.593	49.950	82.140	48.913	87.533	80.340

In access time, the MDSA has the lowest delay back in accessing clustered servers. With the increase of the numbers of CSs, there is an increase of delay in the response time using sequential method. Although the random method resulted in the desired decreasing linear pattern, it was not as pronounced as with the MDSA method.

The results of the single query process are similar to access time. At first CS, both random and MDSA share similar characteristics. The random method shows erratic results. It is the effect of random CPU utilization in every CS and random selection in connecting CS. The sequential method appears to deliver a nonlinear trend result which depicts a higher return on response time for each additional CS.

Table 2. Percentage time difference on MDSA as compared to sequential and random search method during random CPU utilization

No. of Servers	Access Time		Single Query		Multiple Queries	
	With Respect to Sequential Method	With Respect to Random Method	With Respect to Sequential Method	With Respect to Random Method	With Respect to Sequential Method	With Respect to Random Method
1	-0.200	2.432	42.928	31.453	31.368	15.345
2	49.950	49.900	41.001	18.270	52.819	40.372
3	66.589	0.000	53.697	-61.079	58.138	67.128
4	74.919	66.556	68.006	37.607	72.859	32.351
5	49.950	79.980	69.130	69.396	47.460	36.991
6	74.919	74.919	40.047	63.768	70.426	56.211
7	85.698	79.952	74.435	70.114	72.968	6.340
8	87.332	-1.096	81.051	62.572	86.086	82.062

In multiple queries, the MDSA always has the lowest response time from one to eight CSs as compared to sequential and random method. However, the random method has similar characteristics in lower delay in several CSs. The sequential method result is gradually increasing the delay of response time as the number of CS increases.

In general, the MDSA returns a higher percentage time difference over sequential and random method as CS increases . However, there are a number of CS setups where the random method delivers a lower delay in the response time as compared to MDSA in access time and single query. On the other hand, it should be noted that MDSA appears better when multiple queries are used.

4. Conclusion and Recommendations

Using the MDSA method and moving from one CS to eight CSs under a controlled CPU utilization, there is a decrease in delay of response time. The average delay increases from 18% to 87% switching from the MDSA to the sequential method, and by 2% to 87% when switching to the random method.

In random CPU utilization, the MDSA method is more efficient as compared to sequential and random methods. Setting from one up to eight CSs, and switching from the MDSA to the sequential method, the average delay increases from 31% to 87% and when switching to the random method, the average delay increases from 2% to 86%.

With the help of MDSA, there is a decrease in the delay back to distribute requests to a given distributed database node to the originating client.

Clearly, the MDSA method has outperformed the random and even the sequential method. Possible enhancements to the MDSA might include the following methods: (1) doubling the CSs from eight to sixteen, each serving different applications, and dynamically allocating CSs to web server applications as needed, then releasing the CSs to the other allocation server when load increases as web client demand increases, (2) increasing the number of nodes each CSs from 3 to 20 nodes in running the MDSA method and testing random and controlled utilization, (3) increasing the number of clients from 1 to 50 in accessing each CSs, and (4) use more complicated queries that may test the processing time of MySQL Cluster.

5. References

Amiri, A. (2004). A Coordinated Planning Model for the Design of a Distributed Database System. Information Sciences, Volume 164. Numbers 1-4. pp. 229-245.

Anthes, G. (2003). Grids Extend Reach, Computerworld, pp. 29-30.

Brown, C., Guster, D. and Krzenski, S. (2007). Can Distributed Databases Provide an Effective Means of Speeding Up Web Access Times, Journal of Information Technology Management. Volume XVIII, Number 1. pp. 1-15.

Johnson, M. (2003). Gridlock Reality, Computerworld, p 24.

Oracle Corp., (2010). MySQL Cluster for Web and E-Commerce Applications: Growing Revenues and Enhancing Customer Loyalty

Pukdesree, S., Lacharoj, V. and Sirisang, P. (2010). Performance Evaluation of Distributed Database on PC Cluster Computers using MySQL Cluster. Proceedings of the World Congress on Engineering and Computer Science 2010. Volume I.

Silberschatz, A., Korth, H.F., and Sudarshan, S. (2006). Database System Concepts. Fifth Edition. New York, NY: McGraw Hill.