Indian Journal of Science and Technology, *Vol 8(11), DOI: 10.17485/ijst/2015/v8i11/71796, June 2015*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Presentation of a New Approach to the Rapid Failure Recovery in Databases using the Alternative Base Method and Failover Clustering Technique

Mahmood Zamani[1*] and Mohsen Afsharchi[2]

[1]Department of Computer and Electronics, Islamic Azad University of Zanjan, Iran; Zamanioracle@gmail.com
[2]Department of Computer and Electronics, Zanjan University, Zanjan, 45371-38791, Iran

Abstract

Background/Objectives: Database systems are one of the largest communicative interfaces between users and service providing organizations. Generally, all the organizations and companies which deal with sensitive financial data and online customer services, such as banks, the telephone company and their payments need to provide services with high accessibility and maximum efficiency. In order to achieve these objectives, the criterion of high accessibility is investigated in order to provide non-stop services in databases. Since the systems with a high number of users at the level of database conduct a lot of transactions, the running transactions allocate system resources to themselves; as a result, the system is faced with resource constraints and its performance is disrupted. **Methods/Statistical analysis:** In this research, the combination of alternative databases and clustering are used in order to facilitate the recovery and increase the accessibility. Also, job scheduling methods are stated at the time of database failure so that the running clusters can bear the workload of bad clusters. **Results:** The node allocation algorithm is dealt with in this research which is conducted with Weibull distribution. Compared to the random node allocation strategy, the results of simulation indicate that using the above-mentioned algorithm decreases the effect of node failure on system performance in cluster systems in large scale. **Conclusion/Application:** The algorithm decreases the average response time.

Keywords: Clustering, Databases, High Accessibility, Rapid Failure Recovery

1. Introduction

Organizations and companies which deal with customers data are exposed to problems such as disconnection, database server downtime, hard drive failure, or so forth. These problems result in inappropriate services, dissatisfaction, loss of customers, and recession in long term. If the disconnection period increases exponentially, it will have a disastrous impact on business. Therefore, it is essential to investigate the determinative factors of protecting the information and providing the maximum accessibility for the users. In paper [1] presented by Sybase, the need for a high accessibility solution is usually adjusted with business requirements. The author believes that considering a lot of rewards for the customers can improve the business. Moore et al.[2] investigated service providing with high accessibility using three standby methods. They stated the advantages and disadvantages of these three methods with comparison. In paper [3], a considerable improvement was achieved by extension components in high accessibility. In paper [4], Markov modeling approach has been used to achieve high accessibility through clustering method for failure recovery so that the best result can be reached. Paper [5] is about a type of architecture which is based on data redundancy and process redundancy. It is very useful and available in constructing database management systems. Many operations can be done with this architecture, such as keeping various versions and copies of data. In Oracle Data Guard in [6], the objective is to guarantee high accessibility, data protection and their recovery for organizational

*Author for correspondence

data. A complete set of services which include creating, maintaining, managing, and monitoring one or more standby databases for surviving natural disasters and data failure has been investigated in the paper.

This project aims to increase the accessibility to database systems and recovery system failures in the minimum downtime or to run a non-stop recovery so that it relieves the organizations of such problems. In order to fulfill a higher accessibility with available resources, it is necessary to create a synchronized alternative database along with the main database. Clustering the database increases the accessibility as well as increasing system efficiency with proper load distribution. If database samples increase, the perturbation tolerability of them can be increased at OS level (or the virtual machine) and each database node. An algorithm is proposed for this purpose in present research.

The proposed method is investigated in the two following parts. In part 3, the simulations are analyzed, and the conclusion is finally presented in part 4.

2. The Proposed Method of Rapid Failure Recovery in Databases

For the clustered database in a large scale such as a country's banking network, the cluster node failure causes a serious challenge. Although several job scheduling methods are available for improving error tolerability in the clustered database systems, they all present a specific strategy against failure so that the running cluster nodes bear the workload of disabled nodes. These methods require an interaction between reliability and cost. If the precisely temporal and positional information pertaining to the distribution of failing nodes is specified in the real environment, the accessibility is increased by preventing from performing scheduled jobs on failing nodes. Consequently, many researches are continuously conducted on modeling the attributes of disabled nodes throughout the word. The common method of detecting the failure is to use the statistical analysis of recorded errors. The job scheduling process is executed in two stages of job selection and node allocation. In node allocation stage, the node workload information and ignoring the failing node are currently applied.

When one node fails, all processes which are running on it are terminated, and parallel jobs pertaining to it are aborted. Therefore, the failing node can be put into the job cycle by rebooting. The performance of job scheduling system is improved by prioritizing the reliable nodes during the allocation. The strategy applied in this research is a scheduling strategy which is based on the marginal distribution for failing times between failures in one cluster by applying Weibull distribution with a parameter of Shape <1 and is stated through simulating this strategy.

The well-known selecting policies are FIFO, FirstFit, BestFit and so forth. We have used FirstFit policy to manage the waiting input queue. The node allocation policy is a method which selects the most appropriate node among usable nodes for this job. In this research, we consider all nodes to be homogenous and ignore the positional attributes of nodes in the network. The criteria for evaluating job scheduling policy include throughput and average response time. Therefore, an algorithm is proposed for exploitation in this project in which the node results from longer service time, higher reliability and lower probability of failure. The main idea of this method is to select a node with longer servicing among the nodes which are used in the queue. The following inputs are definable in this method:

Queue_SJ: Requested Jobs Queue. The main job information includes job ID, job size or the number of required nodes and job runtime.
Queue_RJ: Running Jobs Queue
Queue_EJ: Ended Jobs Queue
Policy_S: It includes job selection policies which select the appropriate job form Queue_SJ.

FirsFit policy is used in this research. The job selection policy will work under four conditions which are as follows: when a new job arrives; when a job is finished; when a running job has been disconnected and reloaded in Queue_SJ; and when the scheduling cycle starts.

Pool_UN: It is the place where usable nodes are aggregated. A node is usable when it doesn't have a job to run, otherwise it is unusable.
Pool_UUN: It is the place where unusable nodes are aggregated.
Sequence_R: It is a sequence of rebooting events pertaining to nodes sorted by time.

The Time-To-Reboot (TTR) has been modeled less than one in cluster by applying Weibull distribution with the parameter shape.

When a rebooting event arrives for the node, the running job on this node is stopped and reloaded in Queue_SJ. At the same time, all nodes relating to this job come together in Pool_UN.

Uptime_Th: It is the Uptime threshold of node. For all nodes of Pool_UN, only the nodes, Uptime of which are longer that Uptime_threshold can be allocated to job.
Uptime_F: It shows the Uptime of a node which is a function of t and node ID. The value a node's Uptime is considered to be zero in TTR. Uptime function can receive its value from Sequence_R.

It is worth mentioning that the investigated algorithm is as follows:

Algorithm:
Input: Queue_SJ
Queue_RJ
Pool_UN
Pool_UUN
Sequence_R
Uptime_Th
Uptime_F
Begin:
Sort the nodes in Pool_UN according to the node's uptime by decreasing order;
CurrentJob = Queue_SJ[0];
While (CurrentJob!= NULL){
m=the number of nodes whose uptime > Uptime_Th in Pool_UN;
n = CurrentJob size;
if (m > n){
Select the first n nodes from Pool_UN;
Put the selected nodes into Pool_UUN;
Move the CurrentJob from Queue_SJ to Queue_RJ;
CurrentJob begins run;
}
CurrentJob = CurrentJob->next;
}
End.

3. Simulation

The components which have been used in this simulation are as follows:

3.1 Client

The client generates the workload with the regulations and gives new jobs to the job scheduler one by one. The impact of failure on system performance is worse in case of the jobs with longer runtimes. Therefore, these jobs are emphasized on more in the simulation.

A random number is selected out of the interval [100, 1000 m] as the job runtime. Also, another random number which is selected out of the interval [30, 80] is considered as the job size. The period between the jobs follows an exponential distribution, and the average time period between the jobs is 100 ms. The total number of jobs is 1000 ones.

3.2 The Order of Rebooting Node Events

The occurrence of rebooting a node refers to the failure of a node, and the order of rebooting node events is one of the scheduler inputs. Each event includes two attributes: the event time and the node in which the event occurs. When an event named A arrives, the scheduler starts the following operations: acquiring the node N in which the event A has occurred; stopping the job J relating to the node N; transferring the job J from the running jobs queue to the available jobs queue in the scheduler; freeing all the nodes relating to the job J; and starting the process of selecting a job.

We generate a sequence of rebooting events whose response time follows Weibull distribution (0.5, 18000). The simulation time is 200000 minutes. Table 1 indicates all the parameters in brief.

Weibull Probability Density Function:

$$f(x;\lambda,k)=\begin{cases}\frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1}e^{-(x/\lambda)^{k}} & x\geq 0,\\ 0 & x<0,\end{cases}$$

Parameters Scale (Real) $\lambda > 0$

Form (Real) $k > 0$

Base $x \in [0; +\infty)$

Table 1. Parameters

Parameter	Value
Job Size	A Random Number out of [30, 80]
Job Runtime (minute)	A Random Number out of [100, 1000]
Time Period between Jobs (minute)	Exponential Distribution with Coefficient £=0.01
The Number of Jobs	1000
Rebooting Time (minute)	Wiebull (0.5, 18000)
Scheduling Cycle (minute)	50 (minute)
Uptime Threshold	10 (minute)
The Number of Nodes	300, 400, 500, 600, 700, 800

The rebooting time of Weibull distribution scale is 18000 and the form of 0.5 during 20000 minutes.

3.3 Nodes Aggregation

Nodes aggregation indicates the cluster size. In order to obtain the impact of node failures on the cluster system with different scales, we have simulated clusters of different sizes. Table 2 indicates the number of clusters which have been studied and the number of times they were rebooted.

3.4 Job Scheduler

The scheduler completes job selection and node allocation. In this simulation, the scheduler begins under four conditions: when it creates a new job; when it is finished; when rebooting events arrive; and when a scheduling cycle begins. The scheduling cycle is 50 ms. Similar to the workload and the order of rebooting events, the job is simulated under three following conditions: first, when no rebooting event is in the queue and all nodes are uptime all the time; second, rebooting events exist, however, the scheduler doesn't consider them; and third, rebooting events exist, and the scheduler considers them by using the above-mentioned algorithm with an uptime of 10 ms. We can compare the node failure impact on the performance of system with the results of the first and second simulations. Also, the increase in system performance percentage resulting from the algorithm can be observed by comparing the results of the second and third simulation.

The results of simulation under the first condition are indicated in Figure 1 and Table 3.

The results of simulation under the second condition are indicated in Figure 2 and Table 4.

The results of simulation under the third condition are indicated in Figure 3 and Table 5.

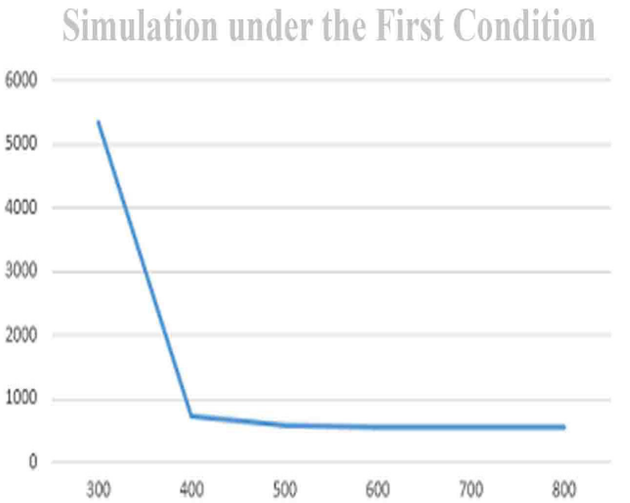

Figure 1. The simulation diagram of numbers collected under the first condition.

The algorithms' throughputs and their intervals are indicated in Figure 4 and Tables 6 and 7.

Comparing the results, it is obvious that the failing nodes have less effect on the throughput. 49% of throughput is decreased for 300 cluster nodes, and the effect of failing nodes on throughput is decreased as the cluster scale increases. Also, the failing nodes have a large effect on the average response time. It means that the average response time is multiplied by 8 in 800 cluster nodes.

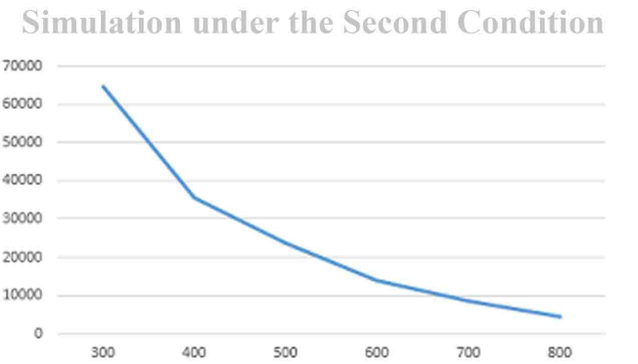

Figure 2. The simulation diagram of numbers collected under the second condition.

Table 2. The number of study clusters and reboots

Nodes	300	400	500	600	700	800
Reboots	2215	2906	3637	4395	5062	5880

Table 3. The average response time (minute) in the first simulation

	300	400	500	600	700	800
Simulation	5353	728	581	555	548	546

Table 4. The average response time (minute) in the second simulation

	300	400	500	600	700	800
Simulation	64690	35679	23637	14008	8597	4392

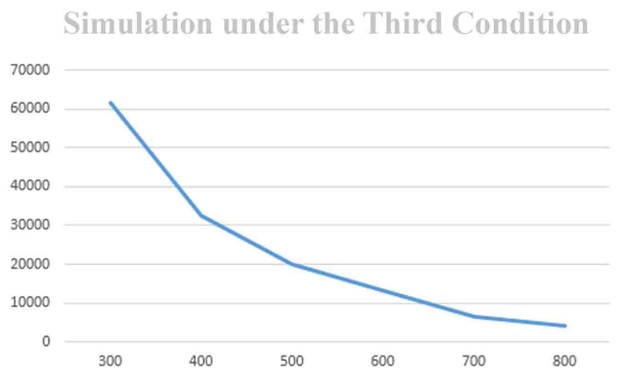

Figure 3. The simulation diagram of numbers collected under the third condition.

Table 5. The average response time (minute) in the third simulation

	300	400	500	600	700	800
Simulation	61812	32434	19949	13266	6401	3997

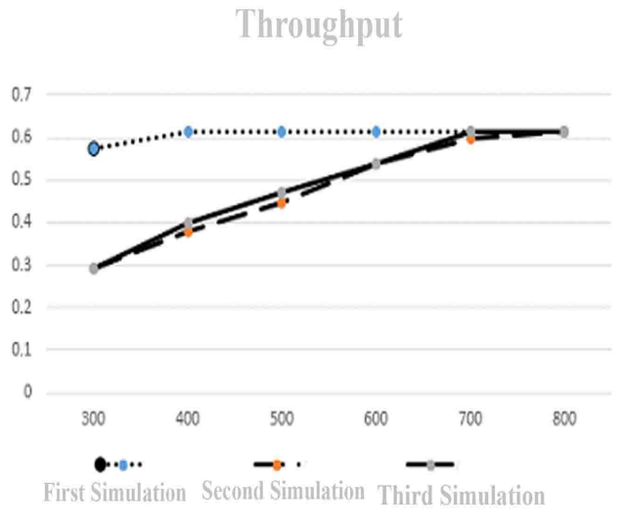

Figure 4. The comparison of throughput diagrams in the simulations.

Table 6. Throughput (the total number of performed jobs/hour)

	300	400	500	600	700	800
First Condition	0.5761	0.6142	0.6142	0.6142	0.6142	0.6142
Second Condition	0.2931	0.3791	0.449	0.5397	0.5999	0.6142
Third Condition	0.2935	0.3985	0.4716	0.5385	0.6138	0.6142

Table 7. The improved performance percentage of algorithm

	300	400	500	600	700	800
Average Response Time	4.45%	9.09%	15.6%	5.3%	25.54%	9.0%

4. Conclusion

The node allocation algorithm is dealt with in this research which is conducted with Weibull distribution. Compared to the random node allocation strategy, the results of simulation indicate that using the above-mentioned algorithm decreases the effect of node failure on system performance in cluster systems in large scale. Also, the algorithm decreases the average response time.

5. References

1. Yeo CS. Cluster computing: high-performance, high-availability, and high-throughput processing on a network of computers. Handbook of Innovative Computing. Springer Verlag; 2005.
2. Moore P. What is the difference between hot, warm and cold disaster recovery? [Internet]. 2011 Jan. Available from:http://www.articlesbase.com/security-articles/what-is-the-difference-between-hot- warm-cold-disaster-recovery-4052188.html
3. Radulescu SB. High availability solution for a transactional database system [Dissertation]. Montreal, Quebec, Canada: Concordia University; 2002 Mar.
4. Lwin TT, Thein T. High availability cluster system for local disaster recovery with Markov modeling approach. IJCSI International Journal of Computer Science. 2009; 6(2).
5. Drake S, Hu W, McInnis DM, Skold M, Srivastava A, Thalmann L, Tikkanen M, Torbjornsen Q, Wolsk A. Architecture of highly available databases. In: Malek M, et al., editor. 2005. p. 1–16. ISAS 2004, LNCS 3335.
6. Oracle: oracle data guard concepts and administration 11g release 2 (11.2) [Internet]. 2011. Available from: http://docs.oracle.com/cd/E14072_01/server.112/e10700/standby.html