Virtual Priority Assignment Policy to Distributed Real-Time Transactions

Sanjeev Kumar Singh, P. K. Singh, R. K. Singh

Abstract— In a distributed environment, tasks generally have processing demands at multiple sites. A distributed task is usually divided into several cohorts, each to be executed in order at some site. This paper deals with an important aspect of distributed real time transaction processing in distributed real-time databases, namely the problem of assigning priorities to transactions. In distributed real time database systems deadline of transactions as well as cohorts affect other transactions during their execution. We perform experiments in three environments: task model, main memory database model and disk resident database model. Our new results show that virtually assigning the priorities of transactions, depending on their behavior, gives a substantial improvement in the number of transactions that meet their deadline in all the three environments.

Index Terms— Distributed Real Time Database System, Deadline Assignment, Priority Assignment, Transaction Scheduling.

———————————— ◆ ————————————

1 Introduction

In DRTDBS, it is common to have transactions with different properties and deadline limit [1, 2]. For some types of transactions, a number of sub-transactions have to be created when the data items required are distributed at different sites. They are called *global transactions*. The others may only require accessing the data items located in their site of origination. They are called *local transactions*. In distributed environment, the processing of a transaction is much more complex than that in single-site. It usually has to create a number of sub transactions (cohorts) to access data objects at different sites [3, 4]. So, the system performance is heavily dependent on the local scheduling of the cohorts at different sites [1, 5, 6, 7, 8, 9]. The deadline constraints of global transactions can be very different from local transactions. Hence, there is need to develop new heuristics for priority assignment of cohorts in order to increase the probability of meeting the deadline.

There are basically two types of distributed transaction execution models; viz., sequential and parallel [7, 10]. In sequential execution model, there can be at most one cohort of a transaction at each execution site, and only one cohort can be active at a time. After successful completion of one operation, next operation in the sequence is executed by the appropriate cohort. At the end of execution of the last operation, the transaction can be committed. In parallel execution model, the coordinator of the transaction spawns all cohorts together and sends them for execution at respective sites [11]. All cohorts then execute in parallel. The assumption here is that the operations performed by one cohort during its execution at one site are independent of the results of the operations performed by some other cohort at some other site. In other words, the sibling cohorts do not share any result among themselves [12]. The transactions can be classified as hard, firm or soft type based on the effect of missing their deadlines [7, 8]. A hard real time transaction must meet its deadline strictly. A missed deadline may result in a catastrophe [7, 9]. A firm real time transaction does not result in a catastrophe, if the deadline is missed [10]. However, the results have no value after the expiry of deadline. Traditionally, in soft real-time database systems, a transaction is considered a monolithic unit of work with a given deadline. In these systems, priorities are assigned to these transactions and the transactions are scheduled based on the basis of their priorities. The priority assignment usually takes into account the deadlines of the transactions because the underlying assumption is that the deadline reflects the urgency of completing the transaction.

In traditional soft real-time applications, a task is considered a single unit of work with a given deadline. The system usually schedules tasks according to their deadlines, with more urgent ones running at higher priorities. Many priority assignments algorithms have been developed over the years and their selection is greatly based on the final application of the database system. The selection of different policy assigning algorithms has different trade-off that the manager of the systems needs to understand and select which one he is willing to have. To that note, in this paper we
will present some of the most used priority assignment algorithms for centralized and distributed systems.

The priority assignment usually takes into account the deadlines of the transactions because the underlying assumption is that the deadline shows the urgency of completing the transaction. Scheduling policies such as Earliest Deadline First (EDF) and Least Slack First (LSF)

• *Research Scholor, Department of Computer Science & Engineering, Uttarakhand Technical University, India, E-mail: sksingh72@gmail.com*

are examples of policies that account for deadlines. The performance implications of time constrained priority assignment policies have been studied in detail in soft real-time database systems in [7, 13]. These studies have concluded that deadline-cognizant priority assignment provides significantly higher performance than priority assignments that ignore deadlines.

In this paper we mainly focus on the development of priority assignment policy that account for the work virtually generated by real time transactions, a comparison of the priority assignment policy with a baseline policy using a real-time database simulator (RADEx) in three environments, in a real-time task model, a main memory database model and a disk-resident database model, priority assignment policies that take into account the dynamic work generated reduce the deadline miss ratio of the executing transactions significantly at the rate of a very small increase in the deadline miss ratio of non-executing transactions when compared to the baseline policy, the identification of the difference between the performance of executing and non-executing transactions provided by the different policies, thereby enabling an implementer to select from various policies depending on the relative importance of the transactions in the system, the identification of differences and reasons for the differences in the relative performance of the policies in the three models that we checked the performance of the priority assignment policy.

The remaining part of our paper is organized as follows: In section II we will discuss the related work done in field of priority assignment policy in RTDBS and in section III we will discuss the virtual priority assignment. The performance evaluation and simulation result will be discussed in section IV and finally in section V we will conclude the paper and give the future scope of this paper.

2 RELATED WORK

Many priority assignments algorithms have been developed over the years and their selection is greatly based on the final application of the database system. The selection of different policy assigning algorithms has different trade-off that the manager of the systems needs to understand and select which one he is willing to have. To that note, in this survey we will present some of the most used priority assignment algorithms for centralized and distributed systems.

One of the earliest and still used policies for priority assignment is the “First-Come-First-Serve” policy. As it sounds, this policy assigns the highest priority to the transaction with the earliest release time [14, 15]. As one can imagine this type of policy have many drawbacks, mainly because it doesn’t take into account the deadline information. Therefore, this policy discriminates against newly arrived tasks that have a closer deadline against tasks that arrived before but might have a farther away deadline. This priority assignment has proven to result in the fewest missed deadline when the workloads are light or moderated. Another one of the most utilized, and first developed policies, is the Earliest Deadline policy. Under this policy the transaction with the earliest deadline receives the highest priority. Once again this policy also has major drawbacks due to the fact that it might assign a higher priority to transactions that are about to missed their deadlines, or have already missed them, denying the utilization of the resources to transactions that might still have an opportunity of meeting their deadlines [14, 15]. One way this problem can be solved is by the utilization of the Not Tardy or Feasible deadline eligibility policy.

Another type of priority assignment that was developed and greatly used is the value-based priority. The main idea behind this algorithm is that the transaction might be assigned a value that will represent its relative importance to the application. The combination of this importance value and the deadline of the transaction will be used to perform the transaction scheduling. In this method an offered value refers to the sum of all the values of the input transactions and a realized value is the sum of all the values of all the transactions that are completed before their deadline. Therefore, the objective in this algorithm is to maximize the realized value. It should be noted that the term value in this algorithm represents the importance weight assigned on a transaction and should not be confused with the priority of a transaction [16]. For this method as for the previous ones, if the deadline is missed no value is realized. As expressed the key idea to this algorithm thus to find the combination between transaction value and deadline that will result in the maximum system-realized value. It should be noted that the task of selecting a function over the other ones is no simple task and it has been shown that the best performance results will depend on a characteristics of a specific application. To this extent several functions have been proposed.

3 VIRTUAL PRIORITY ASSIGNMENT

As discussed above in earliest deadline, value assignment function the priority of transaction is only dependant on the deadline of the transaction. It should be noted that the earliest deadline policy is the function with highest deadline assigned the highest priority to transactions [16]. The main drawback that this priority assignment algorithm has within the new framework is that it doesn’t consider the importance to the system of completing that transaction as well as the transaction that have less deadline to complete their task which causes more number of missed deadline [16].

$$\mathbf{P_T = D_T}$$

Where P_T expresses the priority of the transaction and D_T shows the deadline of a transaction.

Hence to remove the problem discussed above we used the concept of virtual priority which consist of importance value. In this type of priority assignment we consider the each transaction with some importance value V_T. In this we give the highest priority to the transaction which has the shorter deadline of the transaction (D_T) to complete the task and highest importance value (V_T).

$$VP_T = V_T / D_T$$

A transaction with an earlier virtual deadline is served before one with a later virtual deadline. The virtual deadline of a transaction is adjusted dynamically as the transaction progresses, and is computed as a function related to the size of the transaction. In some sense, their approach is to introduce another bias factor counter-attacking the intrinsic bias behavior of earliest-deadline-based scheduling policies. By monitoring the system a parameter in computing virtual deadlines is carefully adjusted in such a way that the linear correlation between miss ratio and transaction size is minimized.

4 PERFORMANCE EVALUATION

Real time active database simulator (RADEx) is written in DeNet languages is used. In our simulation, a small database (200 data items) is used to create a high data contention environment. For each set of experiment, the final results are calculated as an average of 10 independent runs. In each run, 20000 transactions are initiated.

Our primary performance measure is the percentage of missed deadlines (or Miss Percentage, MP) which is defined as the percentage of input transactions that system is unable to complete on or before their deadlines.

Table 4.1: Default Values for the Model Parameters

Parameters	Meaning	Default setting
N_{site}	Number of Site	4
AR	Transaction Arrival Rate	4 Transactions/ Second
T_{com}	Communication Delay	100 ms (Constant)
SF	Slack Factor	1-4 (Uniform Distribution)
N_{oper}	No. of Operations in a Transaction	3-20 (Uniform Distribution)
PageCPU	CPU Page Processing Time	5 ms
PageDisk	Disk Page Processing Time	20 ms
DBsize	Database Size	200 Data Objects/Site
P_{write}	Write Operation Probability	0.60

To investigate the performance of the proposed heuristic and temporary intermediate priority assignment policy, a wide range of transaction size (N_{oper} = 3 to 20) has been used both for global and local transactions.

In figure 4.1 we show the simulation result of task model at normal and heavy load and found that virtual priority have very less miss% in comparison with earliest deadline (ED) and highest value (HV). Hence we can say that virtual priority works well in both the condition.

In figure 4.2 and figure 4.3 we have compare our virtual priority (VP) with the ED and HV in memory resident model at highest communication delay and disk resident database model at minimum communication delay and we have found that our policy outperforms with both the policy to which we compare our priority assignment policy.

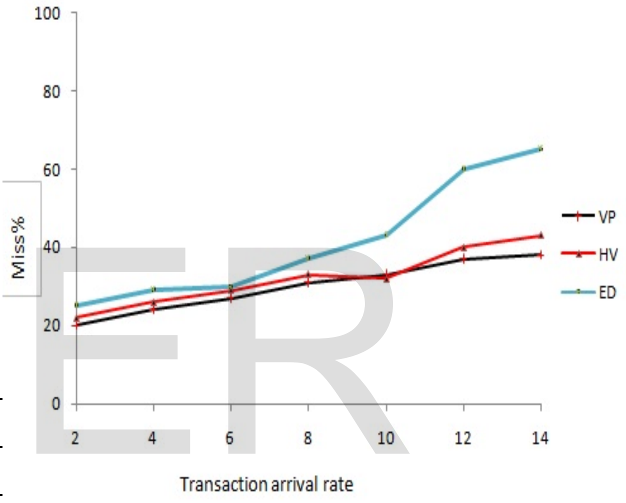

Figure 4.1: Task model at Normal & Heavy load

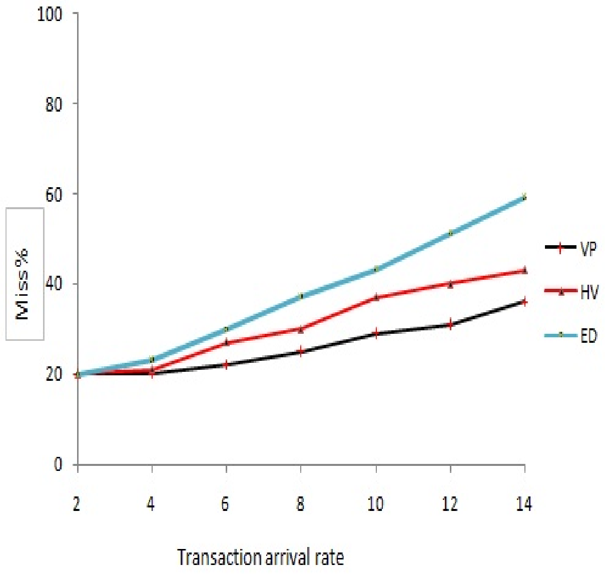

Figure 4.2: Memory resident database model at communication delay=100ms

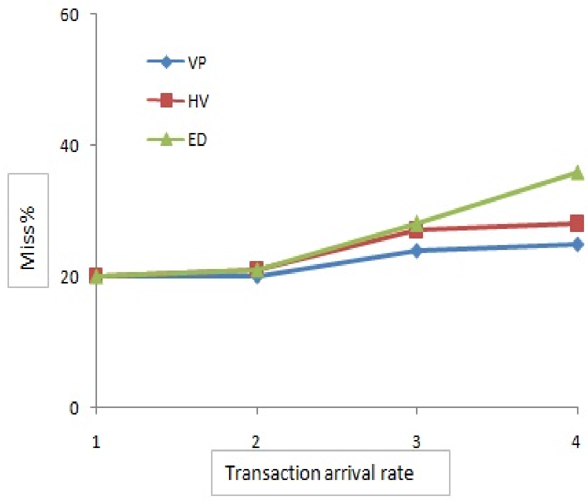

Figure 4.3: Disk resident model at communication delay=0ms

5 CONCLUSION AND FUTURE SCOPE

In this paper we study various priority assignment policy and address the problem with earliest deadline and highest value and also suggest new priority assignment policy with the help of virtual priority concept and analyze the result with the help of simulation and found that our proposed policy works well in various environment in comparison with the policy for that we address the problem. We can also analyze the other policy and can compare the virtual priority and study the behavior in various situations.

REFERENCES

[1]. Kao Ben and Garcia - Molina H., "Deadline assignment in a distributed soft real - time system," Proceedings of the 13th International Conference on Distributed Computing Systems, pp. 428 - 437, 1993.

[2]. Chrysanthis Panos K., Samaras G. and Al - Houmaily Y. J., "Recovery and Performance of Atomic Commit Processing in Distributed Database Systems," Performance of Database Recovery Mechanism, Editors: V. Kumar and M. Hsu, Prentice Hall, pp. 370 - 416, 1998.

[3]. Lam Kam - Yiu, Hung S. L. and Son S. H., "On Using Distributed Real - Time Static Locking Protocols for Distributed Real - Time Databases," Real - Time Systems, Vol. 13, pp. 141 - 166, 1997.

[4]. Lindström Jan and Raatikainen Kimmo, "Using Importance of Transactions and Optimistic Concurrency Control in Firm Real - Time Databases," Proceedings of the 7th International Conference on Real - Time Computing Systems and Applications (RTCSA'2000), Cheju Island, South Korea, December 12 - 14, 2000.

[5]. Ulusoy Ozgur, "Analysis of Concurrency Control Protocols for Real Time Database Systems," Information Sciences, Vol.111, No.1 - 4, 1998.

[6]. Agrawal Divyakant, Abbadi A. El and Jeffers R., "Using Delay Commitment in Locking Protocols for Distributed Real Time Databases," Proceedings of the ACM International Conference on Management of Data (SIGMOD), San Diego, California, pp. 104 -113, June 2 - 5, 1992.

[7]. Agrawal Divyakant, Abbadi A. El, Jeffers R. and Lin L., "Ordered Shared Locks for Real - time Databases," International Journals of Very Large Data Bases (VLDB Journal), Vol. 4, Issue 1, pp. 87 - 126, January 1995.

[8]. Kao Ben and Garcia - Monila H., "An Overview of Real - time Database Systems," Advances in real - time systems, pp. 463 - 486, 1995.

[9]. Ramamritham Krithi, "Real - time Databases," Distributed and Parallel Databases, Special Issue: Research Topics in Distributed and Parallel Databases, Vol. 1, Issue 2, pp. 199 - 226, April 1993.

[10]. Yu Philip S., Wu Kun - Lung, Lin Kwei - Jay and Son S. H., "On Distributed Real - Time Databases: Concurrency Control and Scheduling," Proceedings of the IEEE, Volume 82, No.1, pp. 140 - 157, Jan. 1994.

[11]. Hansen, G.W., Hansen, J.V.: Database Management and Design. Prentice-Hall, India (2000).

[12]. Mohan Chandrasekaran, "An Overview of Recent Data Base Research," Journals of Database, Volume 10, No. 2, pp. 3 - 24, 1978.

[13]. Haritsa Jayant R., Carey M. J. and Livny M., "Data Access Scheduling in Firm Real - Time Database Systems," Journal of Real - Time Systems, Vol. 4, No. 3, pp. 203 - 242, 1992.

[14]. Abbott, R. K., & Garcia-Molina, H. (1988). Scheduling real-time transactions: A performance evaluation. Proceedings of the Fourteenth International Conference on very Large Databases, p1-12, 1988.

[15]. Abbott, R. K., & Garcia-Molina, H. (1992). Scheduling real-time transactions: A performance evaluation. *ACM* Transactions on Database Systems, 17(3), 513-560.

[16]. Haritsa, J., Carey, M., & Livny, M. (1993). Value-based sheduling in real-time database systems.VLDB Journal, 2, 117.