An Improved Transaction Scheduling Policy for Mixed Real-Time Nested Transaction Models

Hong-Ren Chen
Graduate Institute of Technology Development and Communication
National University of Tainan, Tainan 7005, Taiwan R.O.C.
hrchen@mail.nutn.edu.tw

Abstract

In this paper, we study the problem of scheduling transactions in the distributed mixed real-time/non-real-time databases systems, in which both real-time and non-real-time flat/nested transactions exist simultaneously in a multiprocessor environment. *Feedback adjustment policy* (FAP) is proposed to attempt minimizing the number of missed real-time transactions, and reducing the impact on the performance of non-real-time transactions. FAP allocates the processors to both of real-time transactions and non-real-time transactions dynamically based on the real-time ratio for overload adjustment measured statistically by MissRatio. Simulation Results are shown to deliver good performance when an application requires a mixed real-time/non-real-time transaction model.

1. Introduction

In the last decade, most studies in a RTDBS always concern scheduling real-time flat transactions and reducing the *MissRatio* [1-7]. Single type of real-time transactions was only considered in the RTDBS. In practices, not all transactions in advanced database applications have real-time requirements and single level structures. For examples, in Internet travel application, the travel agent transaction may be formed as a nested transaction consisted of three subtransactions. One of subtransactions with real-time constraints is responsible for buying a flight ticket. Others without real-time constraints are executed for lodging and car rentals [8]. In stock trading systems, some of transactions perhaps are nested real-time transactions, such as stock data update. There are maybe non-real-time flat transactions, such as system management proposes [9]. In this paper, we focus on the design of *Feedback Adjustment Policy* (FAP) to accomplish different performance goals in processing real-time and non-real-time transactions at the same time. The performance goal is processing real-time transactions is to minimize the number of deadlines violation while processing non-real-time transactions is to maximize the system throughput.

The remainder of this paper is organized as follows. Section 2 discusses the typical scheduling policies. Section 3 presents FAP and relative processes. Section 4 provides the workload model and performance results. Finally, a conclusion is made in Section 5.

2. Transaction Scheduling Policy

In processing real-time transaction, the real-time scheduler assigns a priority to each (sub)transaction based on the priority assignment policy, such as earliest deadline (ED), highest value (HV), highest reward and urgency (HRU) or flexible high reward for nested transactions (FHRN). For scheduling non-real-time transactions, the traditional scheduling policy provided by general operating system is used, such as first-come first served (FCFS) and shortest-job-first (SJF). In the following, we briefly describe these typical scheduling policies.

2.1 Priority Assignment Policy

The ED policy assigns high priorities to transactions with early deadlines [1]. The disadvantage of ED is that it does not consider the values of transactions. Assigning high priorities to transactions with high values is called HV policy [2]. The HRU policy gives a high priority to a transaction with high value and shortest remaining execution time. It considers the

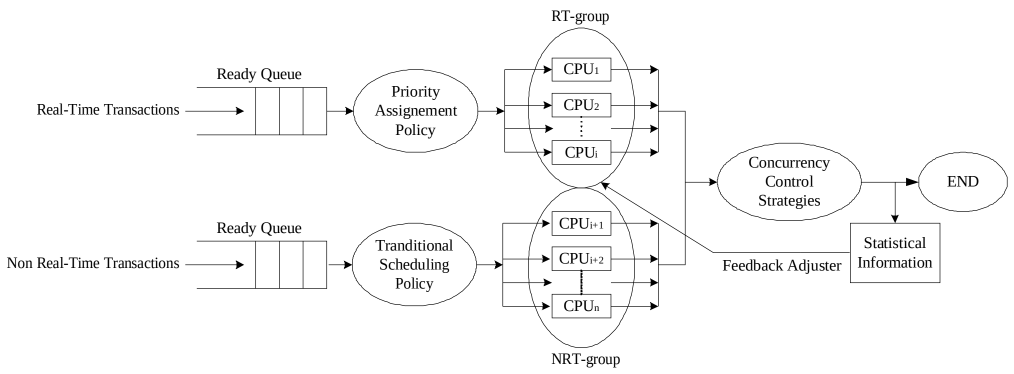

Figure 1. Diagram of FAP [10, 11]

reward ratio of scheduling a transaction and provides an adjustable policy for various system load conditions. Using the concept of the distributed slack time given by Kao and Garcia-Molina [12, 13], and the influence of the shape characteristic for nested transactions [14, 15], FHRN policy is proposed. It considers the reward ratio, reflection of the shape characteristic, distributed slack time and degree of urgency [12].

2.2. Traditional Scheduling Policy

SJF associates with each process the length of the latter's next CPU burst [15]. When the CPU is available, it is assigned to the processes that have the smallest next CPU burst. This paper focuses the minimizing the number of missed real-time transactions, and reducing the impact on the performance of non-real-time transactions. It's only considered in our experiments because its performance.

3. FAP

Attempting to minimize the number of missed real-time transactions and reducing the impact on the performance of non-real-time transactions, FAP is shown in Figure 1 dynamically allocates the processors to both of real-time transactions and non-real-time transactions based on the status of a system. The total CPUs in the system are partitioned into the RT-group that executes the real-time transactions and the NRT-groups that executes the non-real-time transactions by the feedback adjuster. Real-time or non-real-time transactions arriving at the system enter the separate ready queues, and wait to be scheduled by a priority assignment policy or a traditional scheduler for utilizing the CPUs. The number of CPUs allocated to RT-group or NRT-group is adjusted dynamically based on the real-time ratio for overload adjustment measured statistically by MissRatio. Namely, while at most RA percentage of the total CPUs are allocated to the real-time transactions. For instance, suppose the total number of CPUs in the system is 10 and RA is 70%. The number of CPUs for RT-group and NRT-group should get 7 and 3, respectively.

FAP works by the request rule, release rule and adjust rule [16]. The detail functions are described as follows:

Request Rule:

For a real-time (sub)transaction T_i arriving at the sytem, it requests the processors for execution. If there are free processors, the FAP randomly allocates one processor to T_i. In case no free processor exists, the priority of Ti is compared with those transactions in the RT-group to check the possibility of preemption.

Release Rule:

Whenever a processor is released by a (sub)transaction, FAP picks up the (sub)transaction with highest priority in the ready queue of the group to execute on the released processor based on the priority assignment policy or traditional scheduling policy of the corresponding group.

Adjust Rule:

Whenever a number of transactions in an observed period are completed, MissRatio during this period, MR_c, is compared with that of setting for overload adjustment, MR_s.

If $MR_c <= MR_s$, i.e. the status of system is at normal load, no action must be token. Basically, a system is considered as overload if MissRatio exceeds 20% [2]. If $MR_c > MR_s$, it presents the system is under an overload. Let $RA = RA + (MR_c – MR_s)$, which indicates RA is increased by the difference between MR_c and MR_s. On the other hand, the number of processors in RT-group will be increased to get better performance.

4. Performance Evaluation

4.1. Workload Model

This section describes the simulation model was developed by using SIMPACK packages to evaluate the performance of FAP [17]. Table 1 lists the workload model parameters and their base values. The parameters *page_cpu* and *page_io* determine the CPU and disk time needed to access a data page, respectively. The parameter used to model the load of the system is arrival_rate, which specifies the mean rate of transaction arrivals and has a Poisson distribution. In other words, the inter-arrival time of nested transaction is in exponential distributed with mean 1/arrival_rate. *Restart_delay* gives the delay time caused by restarting a transaction. *Write_prob* determines the probability of updating data pages after a transaction has read the data pages. *Sub_trans* signifies the number of subtransactions varying randomly in a (sub)transaction tree. *Tran_size* represents the number of leaf subtransactions in a nested transaction, which is the mean of a uniform distribution varying range between 0.5**tran_size* and 1.5**tran_size*. The parameter *leaf_size* determines the number of operations per leaf subtransaction varying uniformly between 0.25**leaf_size* and 1.75**leaf_size*. The parameter *level_size* represents the depth of a nested transaction tree varying uniformly from 0.25**level_size* to 1.75**level_size*.

The main performance metric is the *MissRatio* as given in [2, 4, 12] and are restated below:

$$MissRatio = \frac{\text{number of transactions missing the deadline}}{\text{total number of submitted transactions}} * 100\%$$

The comparing real-time scheduling policies are ED, HV, HRU, and FHRN+FAP (denoted as FFAP) underlying various conditions. We also concern the variety of response time for non-real-time transactions. SJF and SJF+FAP (denoted as SFAP) were included in this experiment. The concurrency control protocol used is two-phase locking with high priority for nested transactions (2PL-HPN) as shown in [12] because it is a simple and effective protocol for most real-time database researches.

4.2. Basic model

In this experiment, we varied the arrival rate from 20 real-time transactions/second (abbreviated as real-time trans/sec) to 120 real-time trans/sec in increasing steps of 20 to model different system loads. The parameters are set as in Table1 based on the previous studies [1-2, 12, 16, 18]. As shown in Figure 2, the performance order from the best to the worst based on the metrics of MissRatio and LossRatio is FFAP > HRU > HV > ED (i.e., FAP performs the best and ED performs the worst). The excellent performance of FFAP is due to its dynamic allocation of processors to both of real-time transactions and non-real-time transactions based on the system load. Meanwhile, we observe the impact of arrival rate of real-time transactions on response time of non-real-time transaction. Figure 3 that the performance of non-real-time transactions under SFAP is slightly affected and its response time keeps the small value at heavy load of real-time transactions. The effective utilization of rt_ratio makes more real-time transactions meet the deadlines and gets a lower MissRatio.

5. Conclusion

In the past decades, real-time transaction scheduling has been an active research topic. Many previous approaches for scheduling transactions in a RTDBS often assume that all transactions with real-time constraints and having the single level structure. However, the DMRTDBS is an advanced database system in a multiprocessor enviro-

Parameter	Description	Value
System		
num_sites	number of sites in the system	4
num_proc	number of processors in the site	4
page_cpu	CPU time for accessing a data page	0.03 ms
page_io	disk time for accessing a data page	4.8 ms
arrival_rate	the rate of real-time transaction arrivals	50 trans/sec
	the rate of non-real-time transaction arrivals	10 trans/sec
restart_delay	delay time to restart a transaction	5 ms
remote_trans	the ratio of remote transactions in the system	0.3
min_slack	minimal slack factor	2
max_slack	maximal slack factor	8
mean_value	mean value of transaction	100
rt_ratio	adjustment ratio of processors for real-time transactions	70%
nt_period	number of observed transactions in a period	100
Transaction		
sub_trans	number of subtransactions in a (sub)transaction	4
tran_size	number of leaf subtransactions in a nested transaction tree	8
leaf_size	number of operations per real-time leaf subtransaction	4
	number of operations per non-real-time leaf subtran-saction	8
level_size	the depth of a nested transaction tree	4
remote_op	the ratio of remote operations for a remote transaction	0.5
Database & Network		
db_size	number of pages in database	1600 pages
write_prob	write probability for accessing a data page	0.5
transfer_rate	transfer rate of the network	100Mbps
commit_time	commit time for completing a decision phase	40ms

Table 1. Workload parameters and base values.

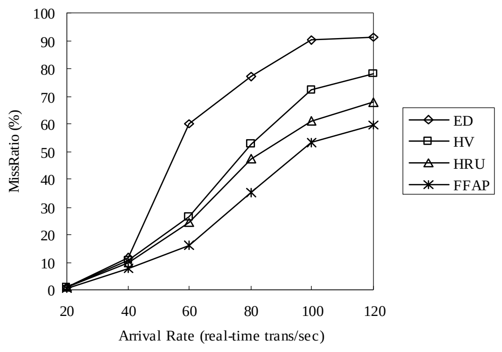

Figure 2. MissRatio for basic model.

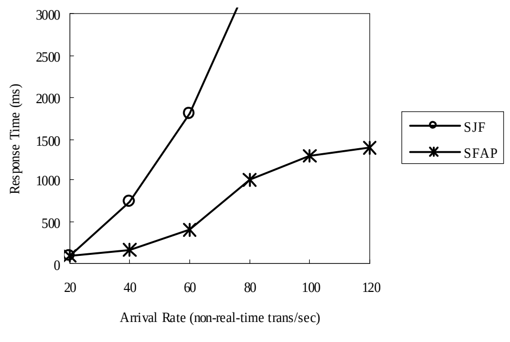

Figure 3. Response time for basic model.

ment consisting of both real-time and non-real-time transactions, and its structure is flat or nested type. The techniques proposed for real-time scheduling policies may not be suitable to DMRTDBS due to the existence of non-real-time transactions.

This paper proposes feedback adjustment policy (FAP) to attempt minimizing the number of missed real-time transactions, and reducing the impact on the performance of non-real-time transactions. The main concept of FAP is utilizing the information recording system load status to allocate processors dynamically to the both of real-time transactions and non-real-time transactions. From simulation results, the performance order from the best to the worst based on the metrics of MissRatio and LossRatio is FFAP > HRU > HV > ED (i.e., FFAP performs the best). The performance of non-real-time transactions under SFAP is slightly affected and its response time keeps the small value at heavy load of real-time transactions.

References

1. Abbott, R., Garcia-Molina, H.: Scheduling Real-Time Transactions: a Performance Evaluation. ACM Trans. Data. Sys, vol. 17, pp. 513-560, 1992.
2. Haritsa, J.R., Carey, M.J., Livny, M.: Value-Based Scheduling in Real-Time Database Systems. VLDB J., vol 2, no. 2, pp. 117-152, 1993.
3. Ulusoy, Ö., Belford, G.G.: Real-Time Transaction Scheduling in Database Systems. Info. Sys., vol. 18, no. 8, pp. 559-580, 1993.
4. Tseng, S.M.: Design and Analysis of Value-Base Scheduling Policies for Real-Time Database Systems. PhD. Thesis, National Chiao Tung University, Taiwan, 1997.
5. Lee, V.C.S, Lam, K.Y., Kao, B.: Priority Scheduling of Transactions in Distributed Real-Time Databases. Real-Time Sys., vol. 15, no. 1, pp. 31-61, 1998.
6. Chen, H.R., Chin, Y.H., Tseng, S.M.: Scheduling Value-Based Transactions in Distributed Real-Time Database Systems. Proc. Int. IEEE Symp. Para. and Dist. Proc, San Francisco, pp. 978-984, 2001.
7. Chen, H.R., Chin, Y.H.: An Adaptive Scheduler for Distributed Real-Time Database Systems. Info. Scie.: an Intl. J., vol. 153, pp. 55-83, 2003.
8. Cram, C.M.:E-Commerce Concepts: Illustrated Introductory. Course Technology, Boston, 2001.
9. Lam, K.Y., Kuo, T.W., Kao, B., Lee, S.H., Cheng, R.: Evaluation of Concurrency Control Strategies for Mixed Soft Real-Time Database Systems. Info. Sys., vol. 27, pp. 123-149, 2003.
10. Ulusoy, Ö.: Processing real-time transactions in a replicated database system. Dist. and Para. Data., vol. 2, no. 2, pp.405-436, 1994.
11. Lee, V.C.S., Lam, K.Y., Hung, S.L.: Impact of High Speed Network on Performance of Real-Time Concurrency Control Protocol. J. Sys. Arch., vol. 42, no. 6-7, pp.531-546, 1996.
12. Chen, H.R., Chin, Y.H. Scheduling Value-Based Nested Transactions in Distributed Real-Time Database Systems. Real-Time Sys., vol. 27, pp. 237-269, 2004.
13. Kao, B., Garcia-Molina, H.: Deadline Assignment in a Distributed Soft Real-Time System. IEEE Trans. Para. and Dist. Sys., vol. 8, no. 12, pp. 1268-1274, 1997.
14. El-Sayed, A.A., Hassanein, H.S., El-Sharkawi, M.E.: Effect of Shaping Characteristics on the Performance of Nested Transactions. Info. and Soft. Tech., vol. 43, no. 10, pp. 579-590, 2001.
15. Silberschatz, A., Galvin, P.B.,Grgne, G. Operating System Concepts, 6rd edn. John Wiley & Sons, 2003.
16. Tseng, S.M., Chin, Y.H., Yang, W.P.: Value-Based Scheduling for Multiprocessor Real-Time Database Systems. IEICE Trans. Info. and Sys. E81-D, no. 1, pp. 137-143, 1998.
17. Fishwick, P.A.: SIMPACK: C-Based Simulation Tool Package Version 2. University of Florida, 1992.
18. Agrawal, R., Carey, M.J., Livny, M. 1987. Concurrency Control Performance Modeling: Alternative and Implications. ACM Trans. Data. Sys., vol. 12, no. 4, pp. 609-654, 1987.