Efficient Priority Assignment Policies for Distributed Real-Time Database Systems

Hong-Ren Chen[1] and Y.H. Chin[2]

[1]Department of Digital Content and Technology
National Taichung University, Taichung 403, Taiwan R.O.C.

[2]Department of Computer Science
National Tseng-Hua University, Hsinchu 300, Taiwan R.O.C.

Abstract: A lot of research works have been done in this filed of real-time database systems to seek for optimizing transaction scheduling. The findings of such studies examining the use of various algorithms of priority assignment policy have been discussed widely. One drawback of these approaches is presenting poor performance due to neglect repetitively missed real-time transactions. In this paper, an improved priority assignment policy named Flexible High Reward with concurrency control factor (FHR-CF) is proposed to reduce the *MissRatio* and *WastedRatio*.

Key-word: priority assignment policy, real-time transaction scheduling, concurrency control factor, real-time database systems

1 Introduction

Researches on real-time database systems have been mounting steadily for a number of decades. There have been efforts to apply these results in many commercial applications, such as real-time data services, wireless sensor networks and traffic information systems [1-4]. Its characteristics not only have to satisfy database consistency, but also must have to consider the time constraint on transactions. In contrast, typical database management systems are only concerned with the data correctness and execution times of transactions. Real-time database systems always deal with the problem of minimizing the *MissRatio* and *LossRatio* for transactions via priority assignment policies and concurrency control mechanisms. In the essential development concept of priority assignment policy, deadline is the time constraint traditionally. Smaller deadlines have higher priority, as with Earliest Deadline (ED) [5]. Some application programs can assign a value to every transaction. When the transaction is completed before the deadline, the transaction will get an actual or a lower value. Hence, priority setting for transactions is based on the value obtained. This is called Highest Value (HV) [6]. We know it is a challenge to satisfy both of deadline and value. Highest Reward and Urgency (HRU) scheduler considers these two factors and make the appropriate combination [7]. We can adjust the weight ratio according to different system loads. With the rapid development of Internet, Flexible High Reward (FHR) for a distributed environment is proposed to reduce the unnecessary waiting time due to communication delay for a remote transaction [8].

Nowadays, numerous researchers have studied the topic of transaction scheduling in real-time databases systems. In spite of that, concurrency control factor (CC factor) is not considered in these researches and lead to poor system performance, especially repetitively executed transactions. In other words, these transactions always

missed deadline at each executing process. Furthermore, most transactions in real-time database systems are routine and executed repeatedly but not necessarily periodically [9]. Hence, the fairness principle designed by CC factor is indicated clearly to avoid starvation occurrences. It considers the following facts: (1) repetitively missed transactions should be assigned higher priority; (2) transactions with larger access cost will get higher priority. More specifically, there may be an opportunity for such transactions to meet their deadlines. In this paper, an improved priority assignment policy called Flexible High Reward with CC factor (FHR-CF) is proposed to reduce the *MissRatio* and lessen the unnecessary restarting for repetitively executing transactions. Simulation results demonstrate that the FHR-CF outperforms current priority assignment policies such as ED, HV and HRU.

The remainder of this paper is organized as follows. Section 2 introduces a distributed RTDBS model. Section 3 describes the FHR-CF in detail. Section 4 presents the simulation model and discusses the performance results. Conclusions are finally made in Section 5.

2 The Model of Distributed Real-Time Database System

In a distributed real-time database system, a communication network interconnects a number of sites as shown in Fig. 1. Each site contains a transaction generator, a transaction manager, a communication interface, a scheduler, a cache manager, and a resource manager. The transaction generator is responsible for generating the workload for each site. The arrivals of data and/or messages at a site are assumed to be independent of arrivals at other sites. Two types of transactions are generated: local transactions and remote transactions. A local transaction accesses data only locally, and a remote transaction accesses data both locally and remotely [10].

The transaction manager is responsible for managing the transaction at multiple sites. When all read-write operations of a transaction have been executed, the transaction enters the commit stage in which an atomic commit protocol is performed. We use the *two-phase commit* (2PC) protocol because it is popular and simple [11]. All sites communicate via data or messages exchange over the communication network. A communication interface at each site is responsible for sending/receiving data or messages to/from other sites. The scheduler assigns a priority to each transaction based on the policy given in Section 3. The scheduler orders access requests for data based on the priority factors. The resource manager provides I/O and CPU services at each site. The global database is a collection of local databases. The problem of access conflicts is resolved by real-time concurrency control protocols involving two-phase locking with high priority (2PL-HP) [5,12].

3 The Proposed Priority Assignment Policy

Priority assignment policies for distributed real-time database systems are developed on the basis of real-time constraints, such as ED, HV and HRU. Generally, such systems will execute the transaction with the highest execution priority assigned to it, based on a particular formula. We briefly describe the following transaction scheduling policy and the notations used are listed below.

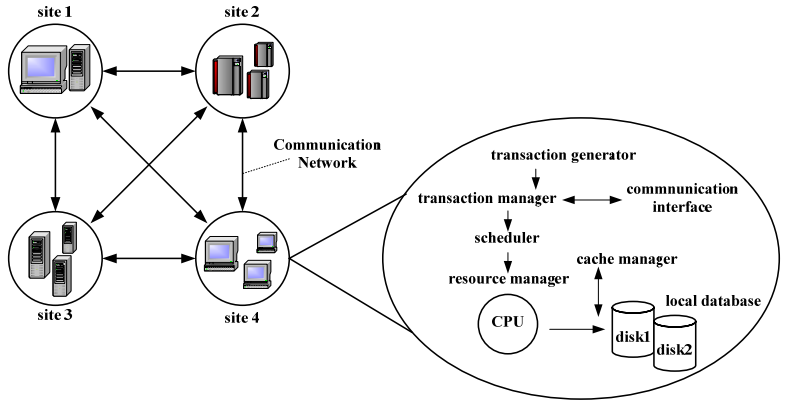

Fig. 1. The framework of a distributed real-time database system [8, 10,12]

T_i	a transaction in the system
$C_{rt}(T_i)$	remaining communication delay of transaction T_i
$E_{rt}(T_i)$	remaining execution time of transaction T_i
$F(T_i)$	the failure ratio of executing transaction Ti
$L_{rt}(T_i)$	remaining locking time of access item for transaction T_i
$P(T_i)$	priority of transaction T_i
$S(T_i)$	slack time of transaction T_i
$V(T_i)$	value of transaction T_i
W	Weight of adjusted transaction's value and urgency

3.1. ED

ED assigns high priorities to transactions with early deadlines [5]. All transactions have the same deadline. The priority assignment formula is given by

$$P(T_i) \leftarrow \frac{1}{D(T_i)}$$

3.2. HV

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 [6]. The priority assignment formula is given by

$$P(Ti) \leftarrow V(T_i)$$

3.3. HRU

In contrast to ED, HV focuses on completing transactions with high values. However, a transaction's urgency is not considered. To eliminate the disadvantage, HRU considers both the deadline and value as design factors. It gives a high priority to a transaction with high value and shortest remaining execution time, and the priority assignment formula is given by:

$$P(T_i) \leftarrow \frac{V(T_i)}{E_{rt}(T_i)} - S(T_i) * W$$

HRU considers the reward ratio of scheduling a transaction and provides an adjustable policy for various system load conditions [7].

3.4. FHR-CF

The above three polices were proposed by [5-7], they are designed for a centralized real-time database environment. Hence, FHR is designed for a distributed RTDBS by extending HRU with communication delay factor to lessen the unnecessary waiting time for a remote transaction. However, these researches usually ignore repetitively missed transactions and lead to poor system performance. The FHR-CF policy based on FHR approach measures this situation and adopts the following actions [8]. The first action is that repetitively missed transactions should be assigned higher priority. Secondly, transactions with larger access cost will get higher priority. There may be an opportunity for such transactions to meet their deadlines. Therefore, we propose the fairness principle designed CC factor in FHR-CF policy to satisfy the higher

failure ratio of transactions with execution histories gets higher priorities, i.e. for a certain transaction the number of uncompleted runs out of all runs of transactions over a period of time. The priority assignment formula is given by

$$P(T_i) \leftarrow \frac{V(T_i)}{E_{rt}(T_i)*L_{rt}(T_i)}*F(T_i) - \frac{S(T_i)}{\text{iif}(C_{rt}(T_i) \neq 0,\ C_{rt}(T_i),1)}$$

4 Performance Evaluation

In the experiment, we varied the arrival rate from 20 real-time transactions/ second (abbreviated as real-time trans/sec) to 100 real-time trans/sec in increasing steps of 20 in order to model different system loads. As shown in Figs. 2a and 2b, the performance order based on the *MissRatio* and *LossRatio* metrics is FHR-CF > HRU > HV > ED (i.e., the FHR-CF performs the best and the ED performs the worst). The excellent performance of the FHR-CF is due to its adjustment policy of fairness principle to repetitively missed transactions. These transactions can get higher priorities based on higher frequent of missing their deadlines or larger cost of disk access. That is, more real-time transactions can be executed completely and responsible for the better system performance. We also observed the impact of the arrival rate of real-time transactions on the utilization of each CPU and disk unit separately. Figs. 3a and 3b shows that under the FHR-CF, the *WastedRatio* increases at a much lower rate than those of the other policies as the load of real-time transactions increases. The effective fairness principle of CC factor in the FHR-CF results in more real-time transactions meeting their deadlines.

5 Conclusion

Previous approaches in real-time database systems usually focused on priority assignment policies to optimize scheduling transactions and minimized the missed real-time transactions. However, most performance studies always neglect the repetitively missed

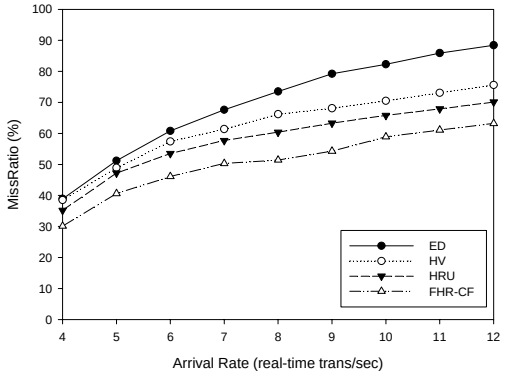

Fig. 2a *MissRatio* for Basic Model

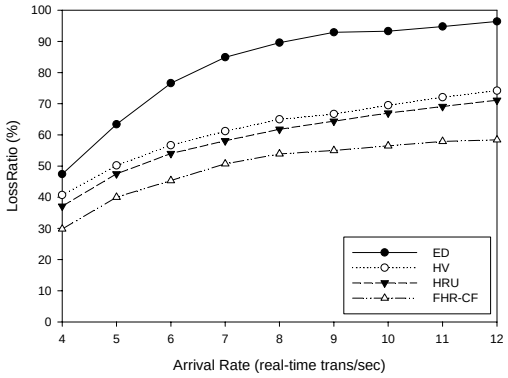

Fig. 2b. *LossRatio* for Basic Model

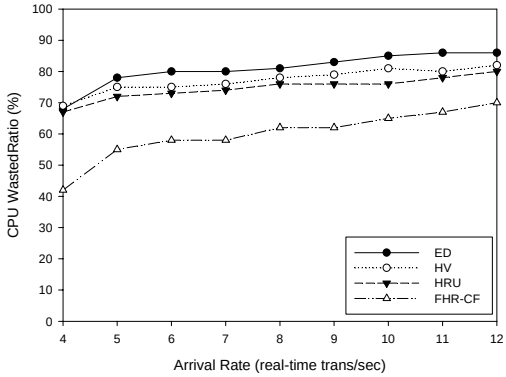

Fig. 3a CPU *WastedRatio* for Basic Model

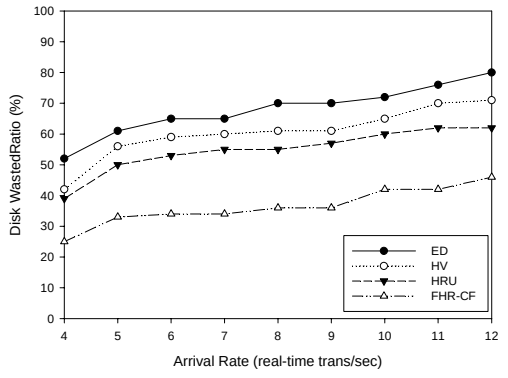

Fig. 3b Disk *WatedRatio* for Basic Model

real-time transactions and lead to the poor performance. Many repetitively executing real-time transactions but not necessarily periodically are used eternally in the practices system. That is, we proposed the priority assignment policy named FHR-CF that considers the fairness principle to eliminate the discriminatory behavior by adjusting the priority with CC factor. The essential work is that satisfy the higher failure ratio of transactions with execution histories gets higher priorities. Through simulation experiments, FHR-CF outperforms current priority assignment polices such as ED, HV and HRU.

References:

[1] K. Ramamritham, S.H. SonL.C. Dipippo, Real-time databases and data services, *Real-Time Systems*, vol. 28, no. 1, 2004, pp. 179-215.

[2] P.F.R. Neto, M. Ligia, B. Perkusich, A. Perkusich, Real-time databases for sensor networks, *Proceeding of the 6th International Conference on Enterprise Information Systems* (ICEIS), 2004, pp. 599-603.

[3] E. Kayan, O. Ulusoy, An evaluation of real-time transaction management issues in mobile database systems, *The Computer Journal*, vol. 42, no. 1, 1999, pp.501-510.

[4] K.Y. Lam, T.W. Kuo, W.H. Tsang, C.K. Law, Concurrency control in mobile distributed real-time database systems, *Information Systems*, vol. 25, no. 3, 2000, pp. 261-286.

[5] R. Abbott, H. Garcia-Molina, Scheduling real-time transaction, *ACM SIGMOD Record*, vol. 17, no. 1, 1992, pp. 513-560.

[6] J.R. Haritsa, M.M. Carey, M. Livny, Value-based scheduling in real-time database systems, vol. 2, no. 2, 1993, pp.117-152.

[7] S.M. Tseng, *Design and analysis of value-based scheduling policies for real-time database systems*, PhD. Dissertation, University of Chiao Tung, Taiwan, 1997.

[8] H.R. Chen, Y.H. Chin, An adaptive scheduler for distributed real-time database systems, vol. 153, 2003, pp.55-83.

[9] E. Dogdu, Utilization of execution histories in scheduling real-time database transactions, *Data & Knowledge Engineering*, vol. 57, no. 2, 2005, pp. 148-178.

[10] Ö Ulusoy, Processing real-time transactions in a replicated database system, *J. Distr. Parallel Database*, 1994, pp. 405-436.

[11] J.R. Haritsa, K. Ramamritham, R. Gupta, The PROMPT real-time commit protocol, *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 2, 2000, pp. 160-181.

[12] H.R. Chen, Y.H. Chin, Scheduling value-based nested transactions in distributed real-time database systems, *Real-Time Systems*, vol. 27, no. 3, 2004, pp. 237-269.

[13] P.A. Fishwick, *SIMPACK: C-Based Simulation Tool Package Version 2*, University of Florida, 1992.

[14] R. Agrawal, M.J. Carey, M. Livny, Concurrency control performance modeling: alternative and implications, *ACM Trans. Database Syst.* Vol, 12, no. 4, 1987, pp. 609-654.

[15] M.J. Carey, M.R. Stonebraker, Performance of concurrency control algorithms for databases management systems, Proceedings of the 10th VLDB Conference, 1984, pp. 107-118.