Algorithm to Process Read-only Transactions in Real-time Environments

VICTOR C. S. LEE  
Department of Computer Science, City University of Hong Kong

KWOK-WA LAM  
Department of Computer Science, City University of Hong Kong

Abstract. In this study, we investigate a different approach to maintaining serializability in real-time database systems (RTDBS) such that concurrency among transactions can be increased. The study is motivated by the dominance of read only transactions (ROTs) in many real-time applications. Given the knowledge about the read/write characteristics of transactions, it can be more efficient and effective to process ROTs separately from update transactions (UTs). In particular, we have devised an independent algorithm to process ROTs while a conventional concurrency control protocol such as optimistic concurrency control (OCC) can be employed to process UTs. Using a separate algorithm to process ROTs can reduce the interference between UTs and ROTs. The undesirable overhead caused by transaction restart and blocking due to concurrency control can be alleviated. Consequently, the timeliness of the system can be improved. The performance of using this approach is examined through a series of simulation experiments. The results showed that the performance of ROTs in terms of miss rate and restart rate is improved significantly while the performance of UTs is also improved slightly. As a result, separate processing of ROTs is a viable approach that achieves better performance and resource utilization than using solely the OCC protocol, one of the best performing protocols in the literature of real-time database.

1. Introduction

Most real-time database applications generate a large proportion of read-only transactions (ROTs). A ROT is a transaction that only reads, but does not write or update any data objects [1]. Examples include information dispersal systems for temporal or time-sensitive information such as stock prices, traffic condition and weather information. In electronic commerce applications, such as stock trading and auctions, it is expected that the number of actual stock buyers or bidders is relatively few with respect to the number of speculators who will also look up (read) the prices frequently. The large population of ROTs makes the processing of ROTs an important performance issue in these applications [2]. Although ROTs can be processed together with update transactions (UTs) by conventional concurrency control protocols (CCPs), in many cases it is more efficient to process ROTs with special algorithms, which take advantage of the knowledge that the transaction only reads [1]. For instance, if ROTs can be processed separately, autonomy between mobile clients and land database server in mobile computing environments can be achieved and the problem of asymmetric communication and limited resources can be eased [3]. As a result, it is important to investigate how ROTs can be processed efficiently...
to the benefit of ROTs themselves and the effective utilization of system resources. It is our belief that processing of ROTs separately from UTs can help to reduce the interference between ROTs and UTs by reducing the undesirable impact of transaction blocking or restart due to concurrency control.

Conventional CCPs do not discriminate between transactions. In other words, ROTs and UTs are processed using a single CCP to ensure the serializability. Previous research studies on concurrency control in real-time database systems (RTDBS) adopted this approach [4–9]. We observe that this approach gives rise to unnecessary blocking or restarts. In locking-based CCPs, long ROTs may hold a large number of locks on data objects for a long time, thus causing UTs to suffer from a long delay. In optimistic CCPs, a committing long ROT may restart a large number of conflicting UTs. The unnecessary blocking or restarts cause transactions missing their deadlines and ineffective use of system resources. In our view, separate processing of ROTs provides greater flexibility and room for resolving the conflicts between ROTs and UTs, thus leading to better performance and less waste of resources.

Serializability is the standard notion of correctness in transaction processing [10] to preserve database consistency. When transactions are processed in a serializable manner, the database is guaranteed to remain in a consistent state. Thus, serializability requires that concurrent executing transactions be scheduled in a serializable manner. Therefore, the objectives of CCPs are to maintain serializability on one hand and to provide maximum concurrency on the other.

Serializability is often a necessary requirement in many RTDBS. However, another prime objective in RTDBS is to meet transactions’ time constraints. This additional objective makes the problem even more complicate. Priority-based scheduling algorithms in real-time systems can be adopted to enforce time constraints of transactions while CCPs in database systems can be used to maintain serializability. Unfortunately, the two mechanisms often operate in an incompatible manner. For instance, transaction blocking mechanism in conventional CCPs may cause undesirable priority inversion problem, in which a high priority transaction is blocked by lower priority transactions [11, 12], in RTDBS. In the literature, a number of real-time CCPs are proposed to relax the strictness of serializability. Some of these studies proposed that serializability could be sacrificed to satisfy the time constraint of transactions [13–16]. They allowed non-serializable execution of transactions and tolerated some temporary and bounded inconsistencies [14, 17–19] such that a timely response to the environment could be provided. For certain real-time applications without the requirement of strict serializability, this approach may be a feasible alternative. However, in many other real-time applications, both timely response and database consistency are of paramount importance such that this approach is rendered inapplicable since even bounded inconsistancy is not acceptable to these applications.

2. Related Work

Most of the real-time CCPs are based on locking or optimistic approach. In locking-based approach such as two-phase locking with high priority (2PL-HP), there are two ways to