Chapter LXXIX
MECP:
A Memory Efficient Real Time Commit Protocol

Udai Shanker
M.M.M. Eng. College, India

Manoj Misra
IIT Roorkee, India

Anil K. Sarje
IIT Roorkee, India

INTRODUCTION

Important data base system resources are the data items that can be viewed as logical resource, and CPU, disks and the main memory which are physical resources [Garcia-Molina et al. 1992]. Though, the cost of main memory is dropping rapidly and its size is increasing, the size of database is also increasing very rapidly. In real time applications, where the databases are of limited size or are growing at a slower rate than memory capacities are growing, they can be kept in the main memory. However, there are many real time applications, which handle large amount of data and require support of an intensive transaction processing. The amount of data they store is too large (and too expensive) to be stored in the main memory. Examples include telephone switching, satellite image data, radar tracking, media servers, computer aided manufacturing etc. [Ramamritham, 1993]. In these cases, the database can not be accommodated in the main memory easily. Hence, many of these types of database systems are disk resident. The buffer space in the main memory is used to store the execution code, copies of files, data pages and any temporary objects produced. The buffer manager controls the main memory and the availability of main memory space affects transaction's response time [Garcia-Molina et al. 1992]. Before starting the execution of a transaction, buffer is allocated for the transaction. When the memory is running low, a transaction may be blocked from execution. The amount of memory available in the system thus limits the number of concurrently executable transactions. In the large-scale real time database systems, the execution of the transaction will be significantly slowed down, if available main memory is low. When the total maximum memory requirement of the admitted transactions exceeds the available memory, distributed real time database systems (DRTDBS) must decide how much memory should be given to each transaction. This decision must also take into account the transactions' timing requirements to ensure that the transactions receive their required resources in time to meet their deadlines. In addition, the effectiveness of memory allocation in reducing individual transaction's response time should be considered, so as to make the best use of the available memory. Therefore, it is important for the database designer to develop memory efficient protocols, so that more number of transactions can be executed concurrently at any time instant. In this work, design of a distributed commit protocol which optimizes memory usage has been presented.

BACKGROUND

The development of commit protocols for the traditional database systems has been an area of extensive research in the past decade. However, in case of distributed real time commit protocols, very little amount of the work has been reported in the literature. The real time commit protocol Permits Reading of Modified Prepared-Data for Timeliness (PROMPT) and deadline-driven conflict resolution (DDCR) commit protocol were proposed by Gupta et al. and Lam et al. respectively [Lam et al. 1999; Haritsa et al. 2000]. Based on the concepts of PROMPT and DDCR, Biao Qin and Yunsheng Liu proposed a new commit protocol double space commit (2SC) [Qin et al. 2003]. All the above protocols consume considerable amount of main memory for maintaining the intermediate temporary records created during the execution of transactions. In PROMPT, the lender maintains extra data structures to record the types of dependencies of its borrowers and DDCR uses more than one copy of the data items (i.e. before, after and further).

All of this not only requires extra memory but also creates additional workload on the system. Furthermore, the locking scheme used by PROMPT and 2SC protocols specifies the lock held by the lender only and these protocols either use read-before-write model or write only (blind write) model. The effect of using both models collectively has not been investigated in any previous work. So, another significant difference between our work and the works reviewed above is that we have considered both blind write (a read is not performed before the data item is written) as well as update (read-before-write). A blind-write model is not unrealistic [Burger et al. 1997] and it occurs in real life information processing for example, recording and editing new telephone numbers, opening new accounts, changing addresses etc. There are also many applications such as banking, intelligent network services database etc. where we need write without ever read model.

MECP

Here, a memory efficient commit protocol (MECP) has been proposed after redefining all kind of dependencies that may arise by allowing a committing cohort to lend its data to an executing cohort under both update (read-before-write) model and blind write (write not ever read) model. A new locking scheme has also been proposed for MECP, in which a lock not only shows the lock obtained by the lender but also the lock obtained by the borrower [Shanker et al. 2005a].

A database is considered as being made up of a set of data items associated with lock variable. The proposed memory efficient commit protocol (MECP) uses a new locking scheme which reduces the need for large number of temporary

Related Content