Available online at www.sciencedirect.com
SCIENCE DIRECT
Journal of Systems Architecture 52 (2006) 373–393
JOURNAL OF SYSTEMS ARCHITECTURE
www.elsevier.com/locate/sysarc

Deferred locking with shadow transaction for client–server DBMSs

Hyeokmin Kwon [a,*], Songchun Moon [b]

[a] Department of Software, Semyung University, San21-1, Shinwoul-dong, Jecheon-si, Chungbuk 390-711, Republic of Korea
[b] KAIST Graduate School of Management 207-43, Cheongryang, Dongdaemun, Seoul 130-722, Republic of Korea

Received 22 February 2004; received in revised form 5 August 2005; accepted 7 September 2005
Available online 9 November 2005

Abstract

Data-shipping systems that allow inter-transaction caching raise the need of a transactional cache consistency maintenance (*CCM*) protocol because each client is able to cache a portion of the database dynamically. Deferred locking (*DL*) is a new *CCM* scheme that is capable of reducing the communication overhead required for cache consistency checking. Due to its low communication overhead, *DL* could show a superior performance, but it tends to exhibit a high ratio of transaction abort. To cope with this drawback, we develop a new notion of shadow transaction, which is a backup-purpose one that is kept ready to replace an aborted transaction. This notion and the locking mechanism of *DL* have been incorporated into deferred locking with shadow transaction. Using a distributed database simulation model, we evaluate the performance of the proposed schemes under a wide range of workloads.

Keywords: Concurrency control; Cache consistency; Transaction processing; Performance evaluation

1. Introduction

With the proliferation of inexpensive, high-performance workstations and networks, client–server database management systems (DBMSs) have been receiving a lot of interest in distributed environments. Client–server DBMSs can be categorized into two classes [7,12,14]: *query-shipping* and *data-shipping*. Most commercial relational DBMSs are based on a query-shipping approach in which most query processing is performed at the server. In contrast, data-shipping systems transfer data items to clients so that query processing can be performed at clients. Most distributed object-oriented DBMSs [11,16,17] today are based on a data-shipping approach. Since data-shipping systems migrate much of DBMS functions from the server to clients, they can off-load shared server machines. However, they are susceptible to network congestion that can arise if a high volume of data is requested by clients.

In data-shipping systems, inter-transaction caching, where cached contents of clients are retained even across transaction boundaries, is an effective technique for improving the performance. Inter-transaction caching enables client resources such

* Corresponding author. Tel.: +82 43 649 1269; fax: +82 43 649 1278.
E-mail addresses: hmkwon@semyung.ac.kr (H. Kwon), scmoon@kgsm.kaist.ac.kr (S. Moon).

1383-7621/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.sysarc.2005.09.002

as memories and CPUs to be effectively used for database-related processing. This also makes it possible to reduce the network traffic and message processing overhead for both the server and clients. However, multiple copies of a data item inevitably happen to exist because each client is able to cache a portion of the database dynamically. Therefore, inter-transaction caching raises the need of a transactional cache consistency maintenance (*CCM*) protocol that basically possesses both aspects of replica management and concurrency control.

Since the *CCM* scheme employed is considered to be substantially affecting the performance of DBMSs, various algorithms [1,7,9,12,13,15,19,21–23] have been proposed in the literature. Transactional *CCM* schemes must ensure that no transactions that access any stale data copies are allowed to commit. According to their policy for guaranteeing this, *CCM* schemes can be classified into two classes [14]: avoidance-based and detection-based. Avoidance-based schemes ensure that all existing copies at clients are always valid. They usually employ a read-one and write-all (*ROWA*) protocol [4] for replica management. The nature of *ROWA* makes them perform read operations on cached copies efficiently without contacting the server. However, they require a substantial amount of work for ensuring that cached pages are valid whenever an updating transaction attempts to commit or to perform a write operation. In addition, these consistency maintenance actions must be performed at one or more remote clients.

In contrast, detection-based schemes allow stale data copies to reside in client caches, and thus each transaction is required to check the validity of any cached copy that it accesses sometime prior to its commit. In detection-based schemes, these consistency checking actions for a single transaction involve only its origin site and the server. Therefore, they could be more robust against client failure than avoidance-based ones. The adaptive optimistic concurrency control (*AOCC*) [1,15] and caching two-phase locking [7,12] can be categorized in this class. *AOCC* can significantly reduce the number of messages required for consistency checking since each transaction is allowed to access cached copies without any constraint. However, it could induce a high ratio of transaction abort, since transactions may read stale data copies. The caching two-phase locking (*C2PL*), which has been designed on the basis of the primary-copy locking (*PCL*) algorithm [4], could cope with the frequent transaction aborts. *C2PL* presumes that cached copies at client buffers are not valid, thus each client transaction is required to check the validity of cached copies before accessing them.

A *PCL-based* scheme usually has the following desirable properties: (1) it is simple and easy to implement, (2) it could be relatively robust against client failure, (3) it could show a low abort ratio. Despite these potential advantages, many studies [7,12,14] report that *C2PL* shows a low performance mainly due to its overhead associated with message requirements for consistency checking. These observations imply that a *PCL-based* algorithm can be widely adopted if it is refined with a mechanism to reduce the communication overhead. In this respect, we devise a novel *CCM* scheme fundamentally on the basis of the *PCL* algorithm. In the proposed scheme, validity check of cached copies is deferred until a client–server interaction is inevitably required due to a cache miss, hence we call this approach deferred locking scheme, *DL* for short. Although *DL* is a *PCL-based* algorithm like *C2PL*, it differs from *C2PL* in that consistency checking is performed on each cache miss, rather than on each access.

DL is capable of reducing client–server interactions significantly by combining a number of lock requests and a data-shipping request into a single message packet. However, it could induce a high ratio of transaction abort due to its lazy buffer validation. This may not be a problem for some kinds of application domain, where transactions are strictly under program control, and also user-initiated requests can be redone on transaction abort without further input. However, the high abort rate of *DL* could make it unsuitable for interactive application domains since a user is obliged to perform all his/her activities again in case of transaction aborts. For this kind of application domain, a mechanism for minimizing negative impacts of transaction aborts would be very beneficial even if it burdens some extra overheads on system resources. From this perspective, we develop a new notion of shadow transaction, which is a backup-purpose one that is kept ready to replace an aborted transaction. Then, we have incorporated this notion and the locking mechanism of *DL* into deferred locking with shadow transaction, *DL–ST* for short.

The remainder of this paper is organized as follows. We present various *CCM* schemes in Section 2 and then we describe the details of the proposed schemes in Section 3. We present a simulation model