Optimistic Concurrency Control for Replicated Objects *

Kyouji Hasegawa, Takayuki Tachikawa, Hiroaki Higaki, and Makoto Takizawa †
Tokyo Denki University ‡
Email {kyo, tachi, hig, taki}@takilab.k.dendai.ac.jp

5ＡＡ－１０

1 Introduction

The distributed applications are composed of multiple objects $o_1, \ldots, o_n$ which are cooperating by exchanging messages through the communication network. Each object o_i supports abstract operations for manipulating the state of o_i. The objects have to be mutually consistent in the presence of multiple accesses to the objects. In the famous two-phase locking (2PL) protocol [1], the transactions lock the objects before computing the operations on the objects. Most systems adopt the strict 2PL protocol where the locks obtained are released at the end of the transactions in order to resolve the cascading abort. The optimistic concurrency control is discussed to reduce the overhead implied by the locking. Here, the transaction manipulates objects without locking the objects. When the transaction T ends up, T commits unless the objects manipulated by T are accessed by other transactions in the modes conflicting with T. In order to increase the reliability, availability, and performance of the system, the objects in the system are replicated. Here, it is critical to make the replicas of the object mutually consistent. Jing [2] discusses an optimistic two-phase locking (O2PL) method to maintain the mutual consistency among the replicas. In the O2PL, all the replicas are locked by a transaction T in a read (*Rlock*) mode to write the object. When T commits, T tries to convert the *Rlock* mode on the replicas to a write (*Wlock*) mode. If succeeded, T commits. Otherwise, T aborts. The distributed applications are modeled in an object-based concept. The objects support abstract operations. In this paper, each object is locked in an abstract mode corresponding to the abstract operation. The conflicting relation between the lock modes is defined based on the conflicting operations. In this paper, we propose a novel optimistic locking scheme for the replicated objects. The number of replicas to be locked depend on how strong the lock mode of the operation is and how frequently the operation is invoked.

2 System Model

2.1 Objects

A distributed system is composed of multiple objects $o_1, \ldots, o_n$ which are cooperating by exchanging messages in the communication network. The communication network supports every pair of objects o_i and o_j with the reliable, bidirectional channel $\langle o_i, o_j \rangle$.

A transaction T sends a request op_i to an object o_i. On receipt of op_i, o_i computes op_i. Here, op_i may invoke an operation op_{ij} on another object o_{ij}. o_i sends the response of op_i back to T. T is an atomic sequence of computation operations. T *commits* only if all the operations invoked by T successfully complete. The operation op invoked by T commits only if all the operations invoked by op commit. op is also an atomic unit of computation. Thus, the operations are *nested*. Each object o_i supports a set τ_i of operations $op_{i1}, \ldots, op_{il_i}$ for manipulating o_i. o_i is encapsulated so that o_i can be manipulated only through the operations supported by o_i. An operation op_{ij} is *compatible* with op_{ik} iff $op_{ij} \circ op_{ik}\ (s_i) = op_{ik} \circ op_{ij}\ (s_i)$ for every state s_i of o_j. op_{ij} *conflicts* with op_{ik} ($op_{ij} \rightarrow op_{ik}$) unless op_{ij} is compatible with op_{ik}. If op_{ij} conflicts with op_{ik}, the state obtained by computing op_{ij} and op_{ik} is independent of the computation order. If some operations conflicting with op_i are being computed on o_i, op_i has to wait until the operations complete.

A transaction T manipulates objects $o_1, \ldots, o_n$ by invoking operations $op_1, \ldots, op_n$, respectively. On receipt of the request op_i, o_i is locked in an lock mode $\mu(op_i)$. Here, let M_i be a set of lock modes of o_i. Two modes m_1 and m_2 in M_i are *compatible* with one another if the operations op_1 of mode m_1 and op_2 of m_2 are compatible. Otherwise, m_1 and m_2 conflict. If o_i is locked in a mode m with which $\mu(op_i)$ conflicts, op_i blocks. op_i is computed after o_i is locked in $\mu(op_i)$. After computing op_i the lock $\mu(op_i)$ of o_i is released. Problem is when o_i is released. Here, suppose that op_i invokes op_{ij} on o_{ij} ($j = 1, \ldots, k_i$). There are the following ways for releasing the locks:

[Releasing schemes]

1. Open : o_i is released when op_i completes.
2. Semi-open : $o_{i1}, \ldots, o_{ik_i}$ are released when op_i completes. However, o_i is not released.
3. Close : Every object locked in op_i is not released. Only if T completes, all the objects locked in T are released. □

2.2 Replicas

An object o_i is replicated in a collection $\{ o_1^i, \ldots, o_i^{k_i} \}$ of replicas, where o_i^j is a replica of o_i. Each o_i^j supports the same data and operations as the other replicas. Let $r(o_i)$ be $\{ o_i^1, \ldots, o_i^{k_i} \}$ ($k_i \geq 1$).

Here, suppose that an operation op_i on an object o_i is invoked. Suppose that op_i invokes an operation op_{ij} on an object o_{ij} and op_{ij} further invokes op_{ijk} on o_{ijk}. Here, suppose that o_{ij} is replicated in multiple replicas $o_{ij}^1, \ldots, o_{ij}^{k_{ij}}$. If op_i sends a request op_{ij} to the replicas $o_{ij}^1, \ldots, o_{ij}^{k_{ij}}$, op_{ij} is computed on the replicas. On receipt of op_{ij}, o_{ij}^k computes op_{ij} and then invokes op_{ijk}. Since multiple replicas invoke op_{ijk}, op_{ijk} is computed multiple times in o_{ijk}. If op_{ijk} changes o_{ijk}, the state of o_{ijk} gets inconsistent since op_{ijk} is computed multiple times on o_{ijk}.

In order to resolve the multiple invocations of the replicas, the following invocation rule is adopted;

1. op_{ij} does not invoke any operation; If op_{ij} changes the state of o_{ij}, op_{ij} is computed on every replica o_{ij}^h. Otherwise, op_{ij} is computed in

*分散環境における楽観的同時実行制御
†長谷川 享二, 立川 敬行, 桧垣 博章, 滝沢 誠
‡東京電機大学

one replica o_{ij}^k.

(2) op_{ij} invokes op_{ijk} on o_{ijk}; op_{ij} is invoked on one replica o_{ij}^k or every replica if op_{ij} changes o_{ij} or not like (1). In either case, only one replica o_{ij}^k invokes op_{ijk}. On receipt of the response of op_{ijk} from o_{ijk}, o_{ij}^k forwards it to all the other replicas if op_{ij} changes o_{ij}^k. On receipt of the state for o_{ij}^k, o_{ij}^h ($h \neq k$) changes the state so as to be the same as o_{ij}^k by using the state.

3 Optimistic Object-Based Locking

3.1 Lock modes

First, we discuss the lock modes supported by the object o_i. Before computing op_i, o_i is locked in a mode $\mu(op_i)$ in M_i. Suppose that o_i is locked in a mode m and op_i would be computed on o_i. If $\mu(op_i)$ is *compatible* with m, op_i can be started to be computed on o_i. Otherwise, op_i has to be waited until the lock of the mode m is released. Here, let $C_i(m)$ be a set of modes with which m conflicts.

[Definition] For every pair of modes m_1 and m_2 in M_i, m_1 is *more restricted* than m_2 ($m_1 \succeq m_2$) iff $C_i(m_1) \supseteq C_i(m_2)$.□

[Definition] For every pair of modes m_1 and m_2 in M_i, m_1 is *stronger* than m_2 iff (1) $m_1 \succeq m_2$ or (2) $C_i(m_1) \cap C_i(m_2) \neq \phi$ and $|C_i(m_1)| \geq |C_i(m_2)|$.□

Some mode m_1 may be more frequently used than m_2. Here, let $\varphi(m)$ be the usage frequency of a mode m, i.e. how many operations whose modes are m are issued to o_i for a unit time. The frequencies of the modes in o_i are normalized to be $\sum_{m \in M_i} \varphi(m) = 1$. The weighted strength $||C_i(m)||$ is defined to be $\sum_{m' \in C_i(m)} \varphi(m')$.

[Definition] For every pair of modes m_1 and m_2 in M_i, m_1 is stronger than m_2 on the weight ($m_1 \succ \succ m_2$) iff (1) $m_1 \succeq m_2$ or (2) $C_i(m_1) \cap C_i(m_2) \neq \phi$ and $||C_i(m_1)|| \geq ||C_i(m_2)||$. □

3.2 Equivalent class

We partition the set π_i of opertions of o_i into groups which are composed of operations interreleted.

[Definition] For every pair of operations op_1 and op_2 in π_i, op_1 and op_2 are interrelated ($op_1 \sim op_2$) iff

(1) op_1 and op_2 conflict or

(2) for some operation op_3 in π_i, $op_1 \sim op_3 \sim op_2$.□

Here, "$\sim$" is reflexive and symmetric. Hence, "$\sim$" is equivalent. π_i is partitioned into the equivalent classes be using "$\sim$". Here, let $R_i(op_1)$ denote an equivalent class $\{op_2 \mid op_1 \sim op_2$ in $\pi_i\}$ of op_1, i.e. for every op_2 in $R_i(op_1)$, $R_i(op_1) = R_i(op_2)$. If $R_i(op_1) \neq R_i(op_2)$, op_1 and op_2 are not inter related, i.e. op_i and op_2 are compatible in o_i.

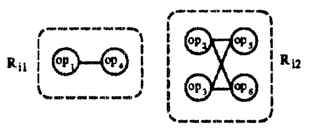

Figure 1: Interrelated operations.

Suppose that an object o_i supports six operations $op_1, \ldots, op_6$ in π_i[Fifure 1]. Suppose that there is a following conflicting relation, i.e. $op_1 \leftrightarrow op_4$ (op_1 conflicts with op_4), $op_2 \leftrightarrow op_5$, $op_2 \leftrightarrow op_6$, $op_3 \leftrightarrow op_5$ and $op_3 \leftrightarrow op_6$. $R_{i1} = R_i(op_1) = R_i(op4) = \{op_1, op_4\}$. $R_{i2} = R_i(op_2) = R_i(op_3) = R_i(op_5) = R_i(op_6) = \{op_2, op_3, op_5, op_6\}$. π_i is partitioned into two equivalent classes R_{i1} and R_{i2}.

We can consider the locking scheme for each class independently of the other classes in o_i. Here, suppose that there is an equivalent class R_{ij} in o_i ($j = 1, \ldots, k_i$).

[Definition] For every pair of operations op_1 and op_2 in each equivalent class R_{ij}, op_1 and op_2 are at the same level ($op_1 \equiv op_2$) iff op_1 and op_2 are compatible and $C_i(op_1) = C_i(op_2)$. □

Here, let $\varphi(op_{ij})$ denote a frequency that op_{ij} is invoked in o_i. Here, the frequencies are normalized to be $\sum_{j=1}^{l_i} \varphi(op_{ij}) = 1$. Each equivalent class R_{ij} can be reduced as follows :

(1) Let $S_{ij}(op)$ be a set of operations which are at the same level op in R_{ij}, i.e. $\{op' \mid op \equiv op'\}$.

(2) All the operations in $S_{ij}(op)$ are replaced with a virtual operation op.

(3) $\varphi(op)$ is given as $\sum_{op' \in S_{ij}(op)} \varphi(op')$.

3.3 Locking protocol

Suppose that a transaction T invokes an operation op_{ij} on o_i. First, some number of the replicas in $r(o_i)$ are locked in a mode $\mu_1(op_{ij})$ which is not stronger than the mode $\mu(op_{ij})$. Let $f_1(op_{ij})$ ($\leq k_i$) be the number of the replicas locked by op_{ij} before op_{ij} is computed. If all of $f_1(op_{ij})$ replicas cannot be locked, op_{ij} is aborted. If all of $f_1(op_{ij})$ replicas are locked, op_{ij} is computed on the replicas as presented before. When T would commit, some number $f_2(op_{ij})$ of the replicas are locked in a mode $\mu(op_{ij})$. $f_1(op_{ij}) \leq f_2(op_{ij})$ and $\mu_1(op_{ij}) \preceq \mu(op_{ij})$. We discuss how to decide the numbers $f_1(op_{ij})$ and $f_2(op_{ij})$ of the replicas to be locked and the lock mode $\mu_1(op_{ij})$. The more replicas are locked, the more communication and computation are required. Hence, the more frequently op_{ij} is invoked, the fewer replicas are locked. We decide $f_1(op_{ij})$ and $f_2(op_{ij})$ depending on the probability that op_{ij} conflicts with other operations of o_i. Here, suppose that op_{ij} locks $f(op_{ij})$ replicas in $r(o_i) = \{o_i^1, \ldots, o_i^{k_i}\}$. Here, $\varphi(op_{ij})$ is a frequency that op_{ij} is invoked in o_i. Suppose that two operations op_{ij} and op_{ik} are invoked and conflict in o_i. The probability that both op_{ij} and op_{ik} can lock the replicas is given 1 - [$f(op_{ij}) \cdot \varphi(op_{ij})/k_i$] [$f(op_{ik}) \cdot \varphi(op_{ik})/k_i$]. $C_i(op_{ij})$ is a set of operations conflicting with op_{ij} in o_i. Here, the probability that op_{ij} can lock of $f(op_{ij})$ replicas is 1 - $\prod_{op \in C_i(op_{ij})}$ $[f(op) \cdot \varphi(op)/k_i]$.

4 Concluding Remark

This paper discusses the optimistic locking protocol on the replicas of the objects. The objects support more abstract operations them the traditional *read* and *write* operations.

References

[1] Bernstein, P. A., Hadzilacos, V., and Goodman, N., "Concurrency Control and Recovery in Database Systems," *Addison-Wesley*, 1987.

[2] Jing, J., Bukhres, O., and Elmagarmid, A., "Distributed Lock Management for Mobile Transactions," *Proc of the 15th IEEE ICDCS*, 1995, pp. 118-125.