Optimistic Type-based Locking on Mobile Objects [*]

7P－8

Takeaki Yoshida and Makoto Takizawa [†]
Tokyo Denki University [‡]
e-mail{take,taki}@takilab.k.dendai.ac.jp

1 Introduction

Kinds of mobile stations like PDA are available at present. Objects are distributed in not only fixed but also mobile stations. Transactions manipulate multiple objects including mobile objects. While the objects are moving from one location to others, the quality of service (QoS) supported by the objects are changed. The connection is closed by the mobile wireless station in order to reduce the power consumption while the operations issued by the mobile station are being computed, i.e. *disconnected* operations [5]. One technique to compute the disconnected operations is to *cache* data in the fixed station like server to the mobile station. Without communicating with the fixed station, users can manipulate the data cached into the mobile station [1, 3]. [4] discusses the locking scheme based on the optimistic two-phase locking [2] on the replicas and a way to reduce the communication. In this paper, the distributed system is assumed to be composed of objects distributed in multiple stations. Each object supports abstract data and operations for manipulating the data, while only *read* and *write* operations are considered in the other papers [1, 3, 4]. Problem is how to support users with the service required by the users under situations where the objects are moving in the system. In this paper, we would like to discuss how to manage transactions which manipulate mobile and replicated objects, which support nested, abstract operations.

In section 2, we present the system model. In section 3, we discuss how to maintain the mutual consistency among the replicas. In section 4, we present the evaluation.

2 System Model

The distributed system is composed of two kinds of stations, i.e. *fixed* and *mobile* ones. The fixed stations are connected at the fixed location of the network.

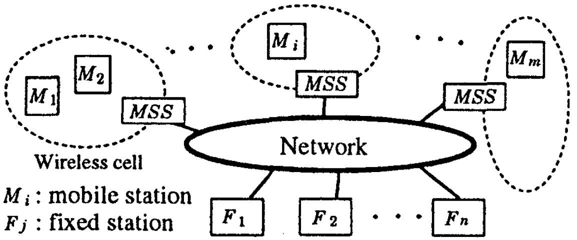

Figure 1: System model

A unit of resource in the system is referred to as *object*, which is composed of abstract data and operations for manipulating the data. Each object o can be manipulated only by the operations supported by o. We assume that each object is stored in one station.

There are two kinds of objects, i.e. *class* and *instance*. The class includes the scheme of the data and the operations for manipulating the data. The instance is composed of the data instance of the scheme and the operations inherited from the class.

Suppose that an object o is replicated into multiple replicas $o^1, \ldots, o^l$ ($l \geq 2$) where each o^i is in a station s_i ($i = 1, \ldots, l$). If the replicas have the same data and operations as o, o is referred to as *fully replicated*.

[Definition] An object o is *mobile* iff the *QoS* supported by o is time-variant. □

The computation of an operation op in an object o may invoke operations in other objects. The computation of op is considered to be *atomic*. The computation of each operation invoked by op is also atomic. Hence, the computation of the operation is considered to be a *nested* transaction.

3 Type Based Optimistic Concurrency Control

3.1 Optimistic locking

The typical scheme to maintain the mutual consistency among multiple replicas is the *read-one* and *write-all* (ROWA) principle. That is, the read operation is issued to one replica while the write operation is issued to all the replicas. In order to reduce the communication overhead, the optimistic approach is adopted. [2] discusses the optimistic two-phase locking (O2PL) protocol. [4] extends the O2PL so as to reduce the communication overhead by avoiding the release of the locks. In the O2PL, one replica is locked by *read* but the replicas are not locked by *write*. When the transaction commits, the replicas updated are locked by *write*. More abstract types of operations are considered in the objects than the *read* and *write* operations.

If an operation op does not change the state of o, op can be computed in only one replica of o. Otherwise, op has to be computed in all the replicas to keep the mutual consistency among the replicas. Problem is the communication overhead since all the replicas have to be locked by the operations whose modes are not minimal.

3.2 Optimistic type-based locking

We adopt the optimistic approach to reduce the communication overhead, named *optimistic type-based locking* (*OTL*). We make the following assumption.

[Assumption] The less restricted the operations are, the more often they are used.□

Each operation op locks some number of replicas in $r(o)$ rather than locking all the replicas. For each operation op_i in o_i, a number $q(op_i)$ is given as follows.

- $q(op_i) < q(op_j)$ if $m(op_i) \prec m(op_j)$.
- $1 \leq q(op_i) \leq l_i$.

---

[*] 移動型オブジェクト上の演算型に基づく楽観的ロック

[†] 吉田 丈成　滝沢 誠

[‡] 東京電機大学

- for every op_j, if $m(op_i) \preceq m(op_j)$, $q(op_i) = 1$.

op_i locks $q(op_i)$ replicas of o_i. For example, suppose that there are five replicas of an object o_i and o_i has three operations op_{i1}, op_{i2}, and op_{i3}. Suppose that $m(op_{i1}) \prec m(op_{i2}) \prec m(op_{i3})$. $q(op_{i1}) = 1$. $q(op_{i2})$ and $q(op_{i3})$ are, for example, given as 2 and 3, respectively. Before computing op_{i2}, two replicas in five ones are locked.

An operation op_i locks an object o_i by the following scheme.

[Locking scheme]

1. Before computing op_i, $q(op_i)$ replicas in $r(o_i)$ are locked in a mode $m(op_i)$. Here, let $s(op_i)$ be a subset of replicas in $r(o_i)$ which are to be locked here.
2. If all replicas in $s(op_i)$ are locked, op_i is computed.
   - (a) If op_i invokes operations in other objects, op_i is computed in one replica in $s(op_i)$.
   - (b) Otherwise, op_i is computed in all the replicas.
3. If some replica in $s(op_i)$ is not locked, op_i aborts.□

We would like to discuss how an operation op invoking op_i commits. One replica o_i^k in $s(op_i)$ plays a role of the coordinator and the other replicas are the participants.

[Commitment]

1. o_i^k sends a *Prepare* message to all the replicas. The participant replica o_i^j which is not locked by op_i, i.e. o_i^j in $r(o_i) - s(op_i)$, is locked in the mode $m(op_i)$ on receipt of the *Prepare* message. If locked, the replica o_i^j sends back *Yes* message to o_i^k.
2. If some replica o_i^j in $r(o_i) - s(op_i)$ is not locked, o_i^j sends *No* to o_i^k.
3. If o_i^k receives *Yes* from all the participant replicas, o_i^k sends *Commit* to all the participants. If o_i^k receives *No* from some participant, o_i^k sends *Abort* to the participants sending *Yes*.
4. If the participant replica o_i^j receives *Abort*, o_i^j abort op_i if o_i^j had computed op_i.
5. If the participant replica o_i^j receives *Commit*, all the replicas in $r(o_i) - s(op_i)$ are locked.
   - (a) Unless op_i invokes operations in other objects, op_i is computed in all replicas in $r(o_i)$ if op_i changes the state, otherwise op_i commits.
   - (b) Otherwise, the state of the replica whose op_i is computed is sent to all the replicas. □

If op_i invokes an operations op_{ij} in another object and op_i is computed in o_i, op_{ij} is computed more than once. In order to avoid the iterated computation, op_i is computed in only one replica, say o_i. In stead of computing op_i in the other replica, the state of o_i is sent to all the other replicas of o_i. If op_i commits, all the locks on the replicas are released.

4 Evaluation

We would like to evaluate the optimistic type-based (OTL) locking scheme by comparing with the traditional read-one and write-all (ROWA) scheme in terms of the number of transactions aborted and the number of lock requests. Here, let o be an object supporting operations $op_1, \ldots, op_h$, which is replicated into replicas $o^1, \ldots, o^l$, i.e. $r(o) = \{o^1, \ldots, o^l\}$. Suppose that $m(op_i) \prec m(op_j)$ $(i < j)$. Let $f(op_i)$ be the probability that op_i is issued to o. Here, $f(op_1) + \ldots + f(op_h) = 1$. $q(op_i)$ denotes the number of replicas to be locked by op_i in the OTL scheme. Here, $q(op_1) = 1$, $q(op_h) = l$, and $q(op_i) \leq q(op_j)$ if $i < j$, i.e. $m(op_i) \prec m(op_j)$. Let A_O be the probability that an operation is aborted in the OTL scheme. A_O is given as $1 - \prod_{i=1}^{h}(1 - f(op_i) \cdot q(op_i)/l) - \sum_{i=1}^{h}[f(op_i)q(op_i)/l \prod_{j=1(j \neq i)}^{h}(1 - f(op_j) \cdot q(op_j)/l)]$. In the OTL scheme, op_i locks $q(op_i)$ replicas. Let A_T be the probability that an operation is aborted in the traditional ROWA way. A_T is obtained by assigning $q(op_1)$ with 1 and $q(op_j)$ with $l (j \geq 2)$ in A_O. Next, let us consider how many lock requests are sent to the replicas. Let L_O and L_T denote the probabilities that each replica is locked in the OTL and traditional ROWA schemes, respectively. L_O is given by $\sum_{i=1}^{h} f(op_i)q(op_i)/l$. L_T is given by $f(op_1)/l + (f(op_2) + \ldots + f(op_k)) = 1 - f(op_1)(l-1)/l$. A_0, A_T, L_0, and L_T are computed for the number l of replicas where $h = 5$, i.e. o has five operations. Here, $q(op_i) = \lceil l/2^{h-i} \rceil (i = 1, \ldots, h)$, $f(op_1) = 0.4, f(op_2) = 0.2, f(op_3) = 0.2, f(op_4) = 0.1, f(op_5) = 0.1$.

Figure 2 and Figure 3 show A_O and A_T, and L_O and L_T for the number l of replicas, respectively. These Figures show that less number of transactions are aborted and less number of lock requests are issued in the OTL scheme than the traditional ROWA.

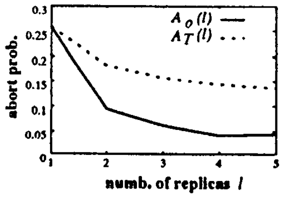

Figure 2: Prob. of abort

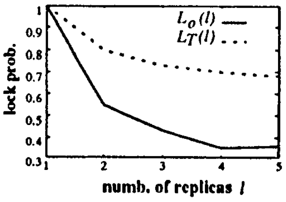

Figure 3: Prob. of lock

5 Concluding Remarks

In this paper, we have discussed how to support nested transactions manipulating replicated and mobile objects in the distributed system. We have discussed the optimistic two-phase locking to maintain the mutual consistency among the replicas. Here, the read-one and write-all principal is extended so that the objects can support more kinds of abstract operations than *read* and *write* and the operations are nested.

References

[1] Barbara, D. and Imielinski, T., "Sleepers and Workaholics: Caching Strategies in Mobile Environments," *Proc. of the ACM SIGMOD*, 1994, pp. 1-12.

[2] Carey, J. M. and Livny, M., "Conflict Detectin Tradeoffs for Replicated Data," *ACM TODS*, Vol. 16, No. 4, 1991, pp.703-746.

[3] Huang, Y., Sistla, P., and Wolfson, O., "Data Replication for Mobile Computers," *Proc. of the ACM SIGMOD*, 1994, pp. 13-24.

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

[5] Kistler, J. J. and Satyanaranyanan, M.: Disconnected Operation in the Coda File System, *ACM Trans. on Database Systems*, Vol. 10, No. 1, pp. 2-25. (1992)