5S−03 Quorum-based Locking Protocol for Replicas in Object-based Systems *

Katsuya Tanaka and Makoto Takizawa †
Tokyo Denki University ‡
e-mail {katsu, taki}@takilab.k.dendai.ac.jp

1 Introduction

Objects are replicated in order to increase the reliability, availability, and performance in distributed object-based applications [3]. In the two-phase locking (2PL) protocol [1], one of the replicas for *read* and all the replicas for *write* are locked. In the quorum-based protocol [2], *quorum* numbers N_r and N_w of the replicas are locked for *read* and *write*, respectively. The subset of the replicas is a *quorum*. Here, a constraint "$N_r + N_w > a$" for the number a of the replicas has to be satisfied. An object is an encapsulation of data and methods for manipulating the data. A pair of methods of an object conflict if the result obtained by performing the methods depends on the computation order. Suppose a pair of update methods t and u are issued to replicas x_1 and x_2 of an object x. The method t may be performed on one replica x_1 and u on another x_2 if t and u are compatible. Here, the state of x_1 is different from x_2. The newest version of x_1 and x_2 can be obtained by *exchanging* methods t and u. The authors [4] discuss a version vector to identify which methods are performed on each replica. However, it implies larger overhead. In this paper, we discuss a simpler protocol to exchange methods among replicas.

In section 2, we overview the quorum-based protocol and discuss the exchanging procedure in the object-based systems. In section 3, we evaluate the quorum-based protocol in terms of the number of messages transmitted compared with the traditional quorum-based protocol.

2 Object-Based Quorum Protocol

2.1 Quorum-based Replication

Suppose there are three replicas x_1, x_2, and x_3 of an object x which supports a method t. A transaction issues a request t to a quorum Q_t, i.e. a subset of replicas to which the request t is issued. In the quorum-based protocol [2], *write* and *read* requests are issued to some numbers N_w and N_r of replicas of x, respectively. Here, $N_w + N_r > 3$. Each replica x_i has a version number b_i. b_i is incremented by one each time *write* is performed on x_i. If a request t is issued, a replica x_i whose version number b_i is maximum in a quorum Q_t is found. If the request is *read*, the value v_i of x_i is read. If the request is *write* with a value v, v is written into the replica x_i. The version number b_i is incremented by one. Then, b_i is sent to the replicas in Q_w.

2.2 Extension of quorum concept

A transaction invokes a method t by issuing a request t to an object x. Then, the method t is performed on the object x and then the response is sent back to the transaction. Here, the method t may invoke other methods, i.e. nested invocation. A method t is *compatible* with a method u iff the result obtained by performing t and u on x is independent of the computation order. Otherwise, t *conflicts* with u.

Let us consider a *counter* object c which supports three types of methods *increment* (*inc*), *decrement* (*dec*), and *display* (*dsp*). Suppose there are four replicas c_1, c_2, c_3, and c_4 of the *counter* object c, i.e. the cluster R_c is $\{c_1, c_2, c_3, c_4\}$. According to the traditional quorum-based theory, *inc* and *dec* are considered to be *write* methods because they change the state of the object c. Hence, $N_{inc} + N_{dec} > 4$, $N_{dsp} + N_{inc} > 4$, and $N_{dsp} + N_{dec} > 4$. For example, $N_{inc} = N_{dec} = 3$ and $N_{dsp} = 2$. Since *dsp* conflicts with *inc* and *dec*, $N_{dsp} + N_{inc} > 4$ and $N_{dsp} + N_{dec} > 4$ in our protocol. However, *inc* and *dec* are compatible because the state obtained by performing *inc* and *dec* is independent of the computation order. Hence, $N_{inc} + N_{dec} \leq 4$, e.g. $N_{inc} = N_{dec} = 2$.

Quorums of an object x have to satisfy the following constraint.

[Object-based Quorum (OBQ) constraint] If a pair of methods t and u conflict, $N_t + N_u > a$ where a is the total number of the replicas. □

2.3 Exchanging procedure

A log L_h is supported for each replica x_h where a sequence of update methods performed on x_h are kept in record. Initially, L_h is empty, i.e. $L_h = \langle]$. Suppose a method t is issued to the replica x_h. If t is an update method, t is stored in L_h, i.e. $L_h = \langle t]$. Here, let L_h be a sequence of update methods $\langle t_{h1}, \ldots, t_{hm}]$ ($m \geq 1$). Suppose an update method t is issued to x_h. If t is compatible with every method in L_h, t is enqueued into L_h, i.e. $L_h = \langle t_{h1}, \ldots, t_{hm}, t]$ and then t is performed on x_h. Thus, every pair of methods in L_h are compatible. Suppose t conflicts with some method t_{hf} and is compatible with every method t_{hg} ($g > f$) in L_h. There might be a replica x_k whose log L_k includes some method t_{kj} which is compatible with every method in L_h but conflicts with t, and is not performed on x_h. Such a method t_{kj} is required to be performed before t is performed on x_h. Here, another replica x_k has a log $L_k = \langle t_{k1}, \ldots, t_{kl}]$ and t is issued to x_k. The method t conflicts with some method t_{ku} and is compatible with every method t_{kg} ($g > u$). According to the OBQ property, every pair of methods t_{hi} in L_h and t_{kj} in L_k are compatible. Here, t_{kj} in L_k is referred to as *missing method* for a method t on a replica x_h iff t_{kj} is not performed on x_h and t_{kj} conflicts with t [Figure 1(1)]. Here, every missing method t_{kj} in L_k for the method t is required to be performed on the replica x_h before t is performed. Then, t is performed on x_h. All the methods conflict-

*分散オブジェクト型システムにおけるレプリカ間の一貫性管理方式

†田中 勝也　滝沢 誠

‡東京電機大学

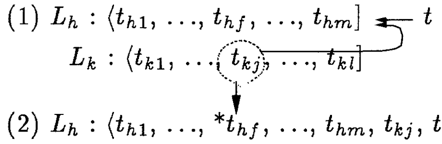
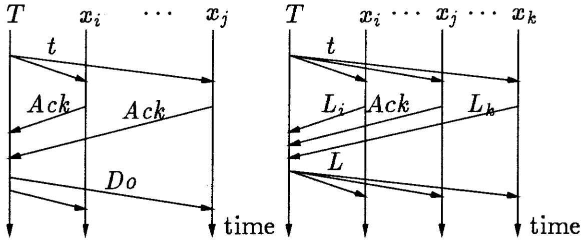

(1) L_h : $\langle t_{h1}, \ldots, t_{hf}, \ldots, t_{hm}]$
t
L_k : $\langle t_{k1}, \ldots, t_{kj}, \ldots, t_{kl}]$
(2) L_h : $\langle t_{h1}, \ldots, {}^*t_{hf}, \ldots, t_{hm}, t_{kj}, t]$

Figure 1: Exchanging procedure.



(1) No conflict (2) Conflict

Figure 2: QB protocol.

ing with t are marked $*$ in L_h and t is enqueued into L_h [Figure 1(2)]. Every pair of unmarked methods in a log are compatible. If an update method t is marked in every log, t was performed on every replica and some method conflicting with t has been performed after t. Hence, t is removed from every log.

A transaction T issues a request t to the replicas in a quorum Q_t:

1. If every method in L_h is compatible with the method t, t is enqueued into L_h if t is an update one and *Ack* (acknowledgment) is sent to T. If T receives *Ack* messages from all the replicas in Q_t, T sends *Do* to the replicas and then t is performed on the replicas.
2. If there is some method in L_h which conflicts with t, L_h is sent to the transaction T.
3. T collects the logs sent by a replica x_h, i.e. $L = L \cup L_h$. If T receives responses from all the replicas, T sends a log L_h' $= \{t' \mid t' \in (L - L_h)$ and t' conflicts with $t\}$, i.e. missing methods for t on x_h to each replica x_h. A method in L_h' is performed on x_h. Then, t is performed on x_h. Every method conflicting with t in L_h is marked.

In Figure 2, a transaction T issues a reuqest t to replicas $x_1, \ldots, x_a$ in the quorum Q_t. Figure 2 (1) shows that the method t is compatible with methods in every log. Figure 2 (2) indicates that t conflicts with methods in some log.

3 Evaluation

We evaluate the QB protocol in terms of messages transmitted compared with the traditional quorum-based protocol. Figure 3 shows a graph where each node shows a method and link between nodes indicates a conflicting relation. For example, *inc* conflicts with *dsp* but is compatible with *inc* and *dec* in Figure 3(1). In the traditional quorum-based protocol, *inc* and *dec* are considered to be *write* and *dsp* is *read*. Figure 3(2) shows a mapping of the conflicting relation of methods to the *read/write* conflicting one. Here, a double circle shows *write* and a single circle indicates *read*. In this evaluation, we assume a method is *write* if the method conflicts with itself. The quorum is decided so as to minimize $N_1 + \cdots + N_l$. For example, $Q_{inc} = Q_{dec} = 2$ and $Q_{dsp} = 3$ for $a = 4$ in the QB protocol. On the other hand, $Q_{inc} = Q_{dec} = 3$ and $Q_{dsp} = 2$ in the traditional protocol.

A sequence of methods is randomly generated for l = number of method types. For each l, possible conflicting relations are obtained. For each conflicting relation, *read/write* conflicting one is obtained as shown in Figure 3.

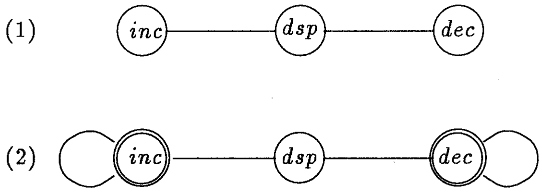


Figure 3: Conflicting relation.

Suppose M_{rw} and M_{me} show the number of messages transmitted in traditional protocol and object-based quorum protocol, respectively. Figure 4 shows $\frac{M_{me}}{M_{rw}}$ for the number of replicas (nr). In this evaluation, we assume there are three types of methods, *inc*, *dec*, and *dsp* and conflicting relations among the methods are obtained as shown in Figure 3. In traditional protocol, "$\frac{nr}{2} + 1$" replicas are locked for each request. In object-based quorum protocol, the quorum for each method is constructed with satisfying the OBQ constraint. Figure 4 shows $\frac{M_{me}}{M_{rw}}$ for the number of replicas. Here, it is shown that only 62 to 87 percent of messages are transmitted in object-based quorum protocol.

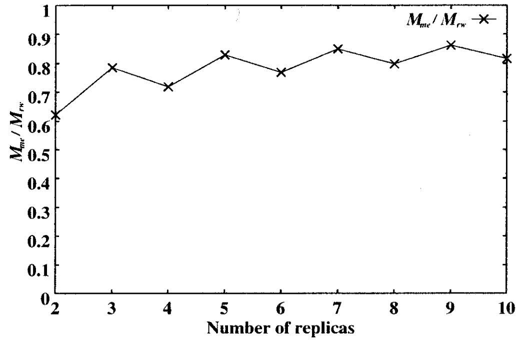


Figure 4: M_{me}/M_{rw}

4 Concluding Remarks

This paper discussed how multiple transactions invoke methods on replicas of objects. The object supports a more abstract level of method than read and write. It is not required to perform every update method instance on the replica which has been performed on the other replicas if the instance is compatible with the instances performed. By using the quorum-based (QB) locking protocol with the exchanging procedure, the number of messages transmitted can be reduced compared with traditional quorum-based protocol.

Reference

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

[2] Garcia-Molina, H. and Barbara, D., "How to Assign Votes in a Distributed System," *JACM*, Vol 32, No.4, 1985, pp. 841-860.

[3] Silvano, M. and Douglas, C. S., "Constructing Reliable Distributed Communication Systems with CORBA," *IEEE Comm. Magazine*, Vol.35, No.2, 1997, pp.56–60.

[4] Tanaka, K., Hasegawa, K., and Takizawa, M., "Quorum-Based Replication in Object-Based Systems," *Journal of Information Science and Engineering(JISE)* Vol. 16, No. 3, 2000, 317–331.