Pseudo-Active Replication of Objects for Heterogeneous Processors *

5 D − 2

Tsunetake Ishida and Makoto Takizawa †
Tokyo Denki University ‡
e-mail{tsune, taki}@takilab.k.dendai.ac.jp

1 Introduction

An approach to making distributed systems fault-tolerant is to replicate objects. The replicas have to be realised in different types of computers. In the active replication, the replicas are computed and communicated in the same synchronous way. The computation speed of the process depends on the slowest replica. In this paper, we discuss a pseudo-active replication where events may not occur simultaneously, not in the same order, and may not occur in the replicas. The succeeding requests can be issued to the replicas if some replies, not necessarily all replies, are received from the replicas without waiting for the completion of the slower replicas.

In section 2, we present the system model. In section 3, we discuss the pseudo-active replication. In section 4, we evaluate the pseudo-active replication.

2 System Model

A distributed application is realised by the cooperation of a *group* G of multiple objects $o_1, \ldots, o_n$. On receipt of a request m from anther object o_j, o_i computes m. If o_i completes m, o_i sends back the response to o_j. A collection $\{o_{i1}, \ldots, o_{il_i}\}$ $(l_i \geq 1)$ of replicas is a *cluster* c_i of o_i. Each computer is characterized in terms of processing speed and reliability level. c_i is *heterogeneous* iff some replicas are in different types of computers. Let $s_{ij}(m)$ and $r_{ij}(m)$ denote the sending and receipt events of a message m in a replica o_{ij}, respectively. Many group protocols are proposed to support a group of multiple processes with the causally ordered delivery of messages. In order to causally deliver messages, each o_{ij} manipulates the vector clock $V = (V_{ih} \mid k = 1, \cdots, n, h = 1, \cdots, l_i)$ [1]. Here, for every pair of messages m_1 and m_2, m_1 causally precedes m_2 $(m_1 \rightarrow m_2)$ iff $m_1.V < m_2.V$. By using the vector clock, the messages received are causally ordered.

Let s_{ij}^0 denote the initial state of o_{ij}. Here, $s_{i1}^0 = \cdots = s_{il_i}^0$. If an event e_{ij}^1 occurs in s_{ij}^0, s_{ij}^0 is transited to the 1st state s_{ij}^1. Here, o_{ij} is represented in a sequence of the events $e_{ij}^1 \circ \cdots \circ e_{ij}^{t_{ij}}$. There occur local events and communication events.

In this paper, the network is assumed to be reliable support messages.

3 Pseudo-Active Replication

In the active replication, every replica is required to do the same computation and communication at the same time.Hence, the computation speed of the cluster c_i depends on the slowest replica in c_i. The response time can be reduced if the computation of c_i completes before every replica. Here, o_h considers that m completes in c_i although o_h does not receive all the responses from c_i.

[Definition] o_{ik} *follows* o_{ij} $(o_{ij} \Rightarrow o_{ik})$ if $m_2^{ij} \rightarrow m_3$ but $m_2^{ik} \not\rightarrow m_3$ for some messages m_3 received and m_2 sent by o_{ij} and o_{ik}. □

Next, we consider how each replica o_{ij} can decide in the distributed way if o_{ij} follows or succeeds other replicas in the cluster c_i. If o_{ij} receives a message m carrying the vector clock $m.V$. o_{ik} knows that $o_{ih} \Rightarrow o_{ik}$ if $m.V_{ik} < m.V_{ih}$. Hence, o_{ij} decides how the replicas are followed on receipt of m according to the following rule.

[Following (F) rule] For every pair of replicas o_{ih} and o_{ik} in c_i, (1) o_{ik} follows o_{ih} if $m.V_{ik} < m.V_{ih}$, (2) o_{ik} and o_{ih} are synchronized if $m.V_{ik} = m.V_{ih}$. □

o_{ik} *follows* o_{ih} *in* o_{ij} $(o_{ih} \Rightarrow_{ij} o_{ik})$ if o_{ij} decides that $o_{ih} \Rightarrow o_{ik}$.

Suppose o_{hl} in c_h sends m_1 and m_3 to o_{ij} and o_{ik} in c_i as shown in Figure 1. o_{hl} sends m_3 on receipt of m_2^{ij} from o_{ij} before receipt of m_2^{ik} from o_{ik}. Hence, on receipt of m_3, $o_{ij} \Rightarrow_{ij} o_{ik}$ and $o_{ij} \Rightarrow_{ik} o_{ik}$.

[Theorem 1] On receipt of a message m, $o_{ij} \Rightarrow_{ih} o_{ik}$ iff $o_{ij} \Rightarrow_{il} o_{ik}$ for every pair of operational replicas o_{ih} and o_{il}. □

If o_{ih} knows that o_{ij} follows o_{ik} by the F rule on receipt of m, o_{ih} receiving m is sure that $o_{ij} \Rightarrow_{ih} o_{ik}$. A message m is *delayed* on o_{ij} if $m.V_{ij} < m.V_{ik}$ for some o_{ik}. If o_{ij} receives a message delayed on o_{ij}, o_{ij} knows that o_{ij} follows some replica.

In the active replication, o_{hl} sends a request message m_1 to o_{ij} and o_{ik}. On receipt of the responses m_2^{ij} and m_2^{ik}, o_{hl} considers that the computation of m_1 completes and sends a request m_3 to o_{ij} and o_{ik}. In the pseudo-active one, o_{hl} does not wait for the responses from all the replicas. Since only stop-fault is assumed to occur, o_{hl} can send m_3 after o_{hl} receives one response from one replica.

Next, suppose that the replicas in c_i receive requests m_1 and m_2. If there is no causally precedence relation between m_1 and m_2, the replicas in c_i may receive m_1 and m_2 in different orders. In the network, every o_{ij} in c_i can take messages in the total order.

The slower replica o_{ik} can catch up with o_{ij} by omitting events not necessarily to occur and by changing the occurrence order of the events.

An event e is an *identity* event in o_{ij} iff $e(s_{ij}) = s_{ij}$ for every state s_{ij} of o_{ij}. An event sequence S is *idempotent* in o_{ij} iff $S \circ S(s_{ij}) = S(s_{ij})$ for every state s_{ij} of o_{ij}. An event sequence S_1 is *absorbed* by S_2 iff $S_1 \circ S_2(s_{ij}) = S_2(s_{ij})$ for every state s_{ij} of o_{ij}.

Two event sequences S_1 and S_2 are *commutative* in o_{ij} iff $S_1 \circ S_2(s_{ij}) = S_2 \circ S_1(s_{ij})$ for every state s_{ij} of o_{ij}. Unless S_1 and S_2 are commutative, S_1 and S_2 *conflict*. For example, two SQL *select* statements are *commutative* but *select* conflicts with *update*. An

*不均質プロセッサにおける疑似能動的多重化
†石田 常竹　滝沢 誠
‡東京電機大学

event sequence S_1 is *equivalent* with S_2 in o_{ij} ($S_1 \equiv S_2$) iff $S_1(s_{ij}) = S_2(s_{ij})$ for every state s_{ij} of o_{ij}.

[Omission rule] Let S_1 and S_2 be event sequences and e be events.

(1) $S_1 \circ e \circ S_2 \equiv S_1 \circ S_2$ if e is an identity.

(2) $e \circ S \circ e \equiv S_1 \circ e$ if e is idempotent and S_1 includes no event conflicting with e. □

[Exchanging rule] If events e_1 and e_2 are commutative and an event sequence S includes no event conflicting with e_1 and e_2, $e_1 \circ S \circ e_2 \equiv e_2 \circ S \circ e_1$ □

We discuss how the slower replicas catch up with the faster ones by using the omission and exchanging rules.

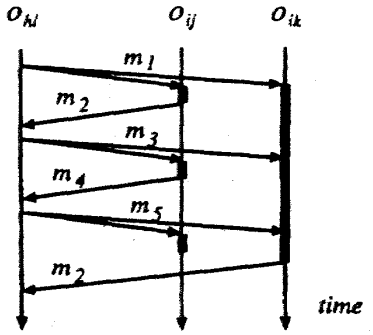

Figure 1: Obsolete message.

[Definition] A message m is *obsolete* in o_{ij} iff (1) m is received but is not delivered to o_{ij}, (2) m is delayed on o_{ij}, and (3) there is some message m' received in o_{ij} such that $m \rightarrow m'$, m' is delayed on o_{ij}, and $m.V_{ij} < m'.V_{ik}$ for some o_{ik}. □

On receipt of a message m, o_{ij} stores m in the causal order in the receipt queue RQ_{ij}. o_{ij} executes the following procedure to check if a top message in RQ_{ij} is obsolete.

[Receipt] A message m arrives at o_{ij}.

(1) m is stored in RQ_{ij} in the causal order by using the vector clock.

(2) If m is delayed on o_{ij}, m is marked *delayed*.

(3) If m is delayed, RQ_{ij} is searched for every *delayed* message m' causally preceding m ($m' \rightarrow m$) in RQ_{ij}.

(3-1) if m' is an identity request, m' is marked *omissible*.

(3-2) if m' is idempotent and there is the same kind of request m'' as m' which precedes m and succeeds m' in RQ_{ij}, m' is marked *omissible* if there is no message between m and m' in RQ_{ij} which conflicts with m'. □

o_{ij} takes the top message m from RQ_{ij}. If m is marked *omissible*, o_{ij} removes m and sends back the dummy response of m with no result.

In Figure 1, suppose m_2 and m_4 are the idempotent requests. m_3 arrives at o_{ik} during the computation of the request m_1. Here, m_3 is delayed on o_{ik} but not obsolete. m_3 is enqueued into RQ_{ik}. Then, the request m_5 arrives at o_{ik}. m_5 is delayed on o_{ik}. The messages in RQ_{ik} are checked by the receipt procedure if they are obsolete. m_3 in RQ_{ik} is obsolete since $m_3 \rightarrow m_5$ and m_5 is delayed. Hence, m_3 is marked *omissible*. After the completion of m_1, i.e. sending m_5, o_{ik} takes m_3 from RQ_{ik}. Since m_3 is marked omissible, o_{ik} omits m_3 and sends back a dummy response of m_3 to o_{hl}. Then, o_{ik} starts to compute m_5 since m_5 is not obsolete while being delayed.

4 Evaluation

The pseudo-active replication $c_i = \{o_{i1}, \ldots, o_{il_i}\}$ supports the same level of reliability as the active one. Let us consider the total computation times of the fastest replica o_{i1} and slowest o_{il_i}. The total time of o_{ij} is defined to be a duration from time when o_{ij} receives the first request until when o_{ij} sends the response of the last request. Let T_A and T_P be the total time of o_{il_i} to compute the requests in the active replication and pseudo-active replication, respectively. Here, suppose 50% of requests are identity ones, and 30% and 20% are idempotent and other requests. o_h sends w (= 100) requests by selecting randomly operations in f_i (= 10) ones. Let δ be a ratio of the propagation delay δ_i among o_h and the replicas in c_i to the processing time τ^i_{i1} of the identity request in o_{i1}, i.e. δ_i/τ^i_{i1}. By the simulation, the total processing times T_P and T_A are obtained. Figure 2 shows T_A/R_A and T_P/R_A with δ = 0.1, 1, 3, 10 for τ_{i1}/τ_{il_i}. The dotted lines show T_P/R_A. From Figure 2 the pseudo-active replication can reduce the total processing time and the number of operations computed in the slower replica. Furthermore, the longer the distance among o_h and the replicas is, the more efficient the pseudo-active replication is.

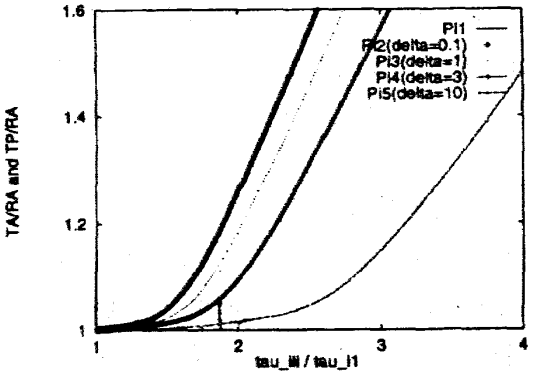

Figure 2: Ratio of total computation time.

5 Concluding Remarks

We have presented the vector clock based method by which each replica can decide how the replica follows others. The slower replicas can catch up with the fastest replicas by omitting the identity and idempotent events and changing the commutative events. We have shown that the pseudo-active replication implies shorter response time and total computation time than the active one while supporting the same level of reliability as the active one.

References

[1] Birman, P. Kenneth and Renesse, V. Robbert, "Reliable Distributed Computing with the Isis Toolkit," *IEEE CS Press*, 1994.

[2] Shima, K., Higaki, H., and Takizawa, M., "Fault-Tolerant Causal Delivery in Group Communication," *Proc. of IEEE ICPADS'96*, 1996, pp.302-309.