Operation Type-based Recovery in Distributed System *

4 O－3

Katsuya Tanaka, Kenji Shima, Hiroaki Higaki, and Makoto Takizawa †
Tokyo Denki University ‡
Email {katsu, sima, hig, taki}@takilab.k.dendai.ac.jp

1 Introduction

In distributed applications like teleconferences, a group of multiple autonomous objects are required to be cooperated by sending messages through communication networks to achieve some objectives.

If an object o is faulty, o and objects which have received messages sent by o have to be *rolled back* to the checkpoint by restoring the information stored in the log taken at the checkpoint and then the computation on o is restarted [1, 2]. Leong and Agrawal [2] present the concept of *significant* messages if the state of an object is changed on receipt of a message m. If o is rolled back, only objects which have received significant messages sent by o are rolled back. In the distributed computation, objects send kinds of messages, i.e. *request*, *response*, and *data* messages. In the significant messages, the transmissions of the request, response, and data messages are not considered. In this paper, we would like to define *influential* messages by taking into account the kinds of messages sent by the objects. Then, we would like to discuss *object-based* checkpoints which can be taken from the object point of view while it may not be consistent.

In section 2, we first present the system model. In section 3, we discuss the influential messages and define the object-based checkpoint. In section 4, we show how the number of checkpoints can be reduced by the object-based checkpoint.

2 System Model

A distributed system is composed of multiple objects interconnected by a communication network. Each object o is defined to be a pair of data structure and a collection P_o of operations. Another object o' can manipulate o only through an operation op in P_o. On receipt of a *request* message m with op from o', op is computed on o. Then, op sends back the *response* message with the result of op. op may invoke operations on other objects, i.e. is *nested*.

For every state s_1 of o, $op(s_1)$ denotes a state s_2 obtained by applying op to s. For every pair of operations op_1 and op_2, $op_1 \circ op_2$ means that op_2 is applied after op_1.

[Definition] Operations op_1 and op_2 of an object o are *compatible* iff $op_1 \circ op_2(s) = op_2 \circ op_1(s)$ for every state s of o. □

op_1 and op_2 *conflict* iff they are not compatible.

An object o supports two kinds of abstract operations, i.e. one changes the state of the object and the other not. op is *stable* if neither op nor any descendant of op changes any object.

Suppose that an operation op_i of an object o_i invokes op_j of o_j. These are two ways to compute op_j. One is *dependent* computation. Here, op_i waits for the completion of op_j after invoking op_j. Otherwise, the computation is referred to as *independent* one.

There are two kinds of messages transmitted among the objects: *control* and *data* messages. The control messages mean requests and responses. After the operations are invoked, they may communicate with other operations by exchanging data messages. Suppose that op_i invokes op_j. If op_i and op_j do not communicate with one another, op_j is *closed* for op_i. Otherwise, op_j is *open* for op_i.

3 Object-based Checkpoints

We assume that each object o_i may stop by fault. o_i takes a local checkpoint c^i where the state of o_i is stored in the log. If o_i is rolled back to c^i, other objects have to be rolled back to the local checkpoints if they had received messages sent by o_i. A collection of the local checkpoints $\langle c^1, \ldots, c^n \rangle$ is a *global checkpoint* c. From here, a term *checkpoint* means a *global* checkpoint. If o_j sends a message m before taking c^j but o_i receives m from o_j after taking c^i, m is an *orphan*. c is *consistent* if there is no orphan [1].

3.1 Dependent invocation

Suppose that an operation op_1^i in o_i invokes op_2^j in o_j. There are four ways to invoke op_2^j: closed dependent, open dependent, closed independent, and open independent computations of op_2^j.

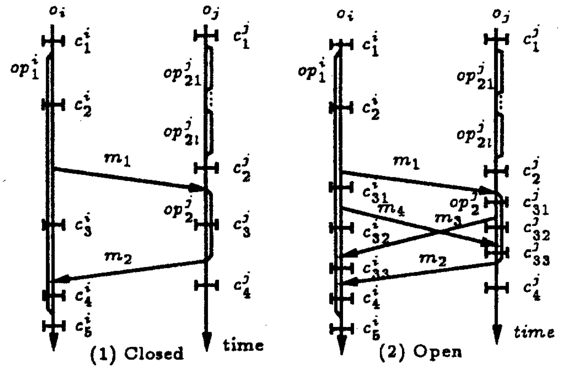


Figure 1: Dependent computation

o_i	o_j	Conditions
c_1^i, c_2^i	c_3^j, c_4^j	op_2^j is stable
c_4^i, c_5^i	c_1^j	op_2^j is stable and no operation in $prec_j(op_2^j, c_1^j)$ conflicts with op_2^j.
	c_2^j, c_3^j	op_2^j is stable

Table 1: O-checkpoints for Figure 1(1)

Here, let $prec_j(op^j, c^j)$ be a set of operations which (1) precede op^j and (2) succeed a checkpoint c^j or are being computed at c^j in o_j. First, we would like to discuss whether each inconsistent checkpoint $\langle c_k^i, c_h^j \rangle$ can be taken or not in Figure 1(1). Here, a checkpoint c is *object-based* (*O-checkpoint*) iff every object can be

*分散システムにおける意味的な復旧方法
†田中 勝也　島 健司　桧垣 博章　滝沢 誠
‡東京電機大学

o_i	o_j	Conditions
c^i_1	$c^j_{31}, c^j_{32}, c^j_{33}, c^j_4$	op^j_2 is stable
c^i_{31}	c^j_4	op^j_2 is stable
c^i_4, c^i_5	c^j_1	op^j_2 is stable and no operation in $prec_j(op^j_2, c^j_1)$ conflicts with op^j_2.
	$c^j_2, c^j_{31}, c^j_{32}, c^j_{33}$	op^j_2 is stable

Table 2: O-checkpoints for Figure 1(2)

rolled back to c and be restarted from c from the object point of view while c may be inconsistent from the definition. $\langle c^i_1, c^j_3\rangle$ and $\langle c^i_1, c^j_4\rangle$ are not consistent. If op^j_2 is stable, the state denoted by c^j_2 is the same as c^j_3 and c^j_4. Since $\langle c^i_1, c^j_2\rangle$ is consistent from the definition, $\langle c^i_1, c^j_3\rangle$ and $\langle c^i_1, c^j_4\rangle$ are object-based because there is no orphan message. Table 1 summarizes the inconsistent but object-based (O) checkpoints.

op^j_2 is open for op^i_1 in Figure 1(2). $\langle c^i_2, c^j_{3h}\rangle$ ($h = 1, 2, 3$) is object-based if op^j_2 is stable. $\langle c^i_{31}, c^j_{33}\rangle$ cannot be taken because m_4 is an orphan. Thus, the data messages are not allowed to be orphans while the control messages could be orphans. Table 2 shows the inconsistent but object-based checkpoints in Figure 1(2).

3.2 Independent invocation

Next, suppose that the invocation of op^j_2 is independent. First, suppose that op^j_2 is closed for op^i_1 [Figure 2(1)]. Table 3 shows the inconsistent but object-based checkpoints.

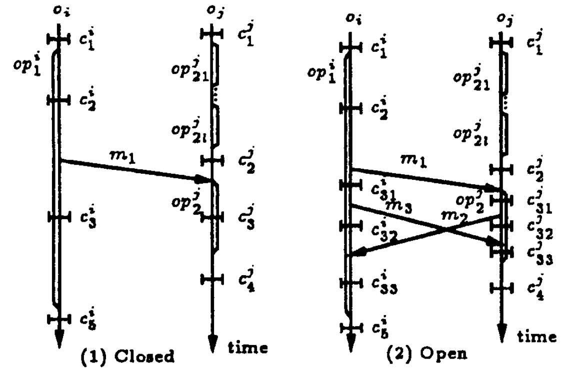


Figure 2: Independent computation

o_i	o_j	Conditions
c^i_1, c^i_2	c^j_3, c^j_4	op^j_2 is stable

Table 3: O-checkpoints for Figure 2(1)

Table 4 shows the inconsistent but object-based checkpoints where op^j_2 is open [Figure 2(2)].

3.3 Influential messages

A message m is *participated* in an operation op if (1) m is a request or response of op or (2) m is a data message received in op. Let $Op(m)$ denote an operation in which m is participated. Following the discussions here, we would like to define the influential messages.

[Definition] Suppose that op^j_2 sends a message m to op^i_1. Let c^i and c^j be checkpoints most recently taken by o_i and o_j, respectively. m is *influential* iff one of the following conditions is satisfied:

(1) If m is a request, $Op(m)$ $(= op^i_1)$ is unstable.
(2) If m is a response, $Op(m)$ $(= op^i_1)$ is unstable and some operation in $prec_i(Op(m), c^i)$ conflicts with $Op(m)$.

o_i	o_j	Conditions
c^i_1	$c^j_{31}, c^j_{32}, c^j_{33}, c^j_4$	op^j_2 is stable
c^i_{31}	c^j_4	op^j_2 is stable
c^i_5	c^j_1	op^j_2 is stable and no operation in $prec_j(op^j_2, c^j_1)$ conflicts with op^j_2.
	c^j_2, c^j_{31}	op^j_2 is stable

Table 4: O-checkpoints for Figure 2(2)

(3) If m is a data message, (3-1) $Op(m)$ $(= op^i_1)$ is being computed, or (3-2) $Op(m)$ is unstable or conflicts with some operation in $prec_i(Op(m), c_i)$. □

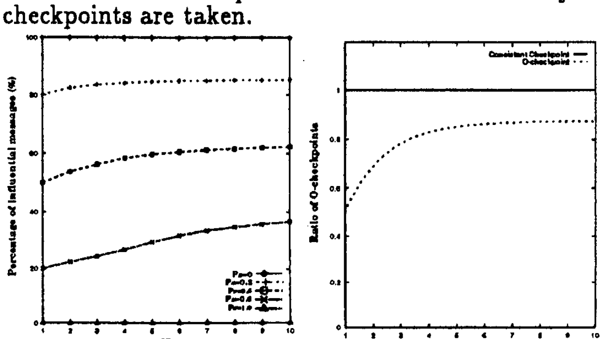
[Definition] A global checkpoint $c = \langle c^1, \ldots, c^n\rangle$ is *object-based* (*O-checkpoint*) iff (1) c is consistent or (2) every influential message is not an orphan at c. □

4 Evaluation

In order to make the evaluation simple, we make the following assumptions:

(1) There are two objects o_i and o_j in the system.
(2) o_i invokes an operation in o_j every u time units.
(3) o_i invokes randomly four kinds of operations, i.e. open dependent, open independent, closed dependent, and closed dependent ones.
(4) In the open invocation of o_j, o_i sends one message to o_j and o_j sends one to o_i.
(5) o_j takes a checkpoint after every cn operations are invoked.

Here, let P_s denote a probability that an operation invoked by o_i is stable. Figure 3(1) shows the percentages of influential messages for the total number of messages which o_j receives. Figure 3(2) shows that the number of checkpoints can be reduced if only O-checkpoints are taken.



(1) Number of influential messages (2) Number of O-checkpoints (P_s = 0.5)

Figure 3: Evaluation

5 Concluding Remarks

We have defined the *influential messages* on the basis of the semantics of request, response, and data messages where the operations are nested. By using the influential message, we have defined the *object-based checkpoint*. We have shown that we can reduce the number of checkpoints to be taken if each object takes only O-checkpoints.

Reference

[1] Chandy, K. M. and Lamport, L., "Distributed Snapshots : Determining Global States of Distributed Systems," *ACM Trans. on Computer Systems*, Vol. 3, No. 1, 1985, pp. 63–75.

[2] Leong, H. V. and Agrawal, D., "Using Message Semantics to Reduce Rollback in Optimistic Message Logging Recovery Schemes," *Proc. of IEEE ICDCS-14*, 1994, pp.227–234.

[3] Tanaka, K. and Takizawa, M., "Distributed Checkpointing Based on Influential Messages," *Proc. of IEEE ICPADS'96*, 1996, pp. 440–447.