ANTECEDENCE GRAPH APPROACH TO CHECKPOINTING FOR FAULT TOLERANCE IN MULTI AGENT SYSTEM

Rajwinder Singh
Department of Computer Science and Enginnering, Chandigarh Engineering College, Landran, Mohali, Punjab, India

Ramandeep Kaur and Rama Krishna Challa
Department of Computer Science, National Institute of Technical Teachers' Training and Research, Chandigarh, India

Keywords: Mobile Agent System, Antecedence Graphs, Fault Tolerance, Checkpointing, Message Logs.

Abstract: Checkpointing has been widely used for providing fault tolerance in multi-agent systems. But the traditional message passing based checkpointing and rollback algorithms may suffer from problems of excess bandwidth consumption and large overheads. In order to maintain consistency of multi agent system, the checkpointing is forced on all participating agents that may result in blocking of agents' operations to carry out checkpointing. These overheads could be considerably reduced if the checkpointing would be forced only on selective agents instead of all agents. This paper presents a low latency, non-blocking checkpointing scheme which marks out dependent agents using Antecedence graphs and then checkpoints are forced on only these agents. To recover from failures, the antecedence graphs and message logs are regenerated and normal operations continued. The proposed scheme reports less overheads and reduced recovery times as compared to existing schemes.

1 INTRODUCTION

A mobile agent (MA) (Nwana, 1996) is a program that represents a user in a computer network and can migrate autonomously from node to node, to perform some computation on behalf of the user. Its tasks, which are determined by the agent application, can range from online shopping to real-time device control to distributed scientific computing. Most of these applications require high degree of reliability and consistency. Therefore, fault tolerance is a key issue in designing an MA system. We consider the scenario of multi-agent system that consists of several collaborating agents and amalgamate the concept of checkpointing and antecedence graphs for fault tolerance in multi agent systems.

As mobile agent systems scale up, their failure rate may also be higher. Several techniques have been proposed for providing fault tolerance in multi-agent systems (Lyu et al, 2004) Rollback recovery could be based on either message logging or checkpointing (Elnozahy, 1999). Log based algorithms require that each agent periodically saves its local state and logs the messages it received after having saved the state. Checkpointing is one of the widely used fault tolerance techniques and may be classified into Synchronous (Meth and Tuel, 2000), Asynchronous (Bhargava and Lian, 1998) and Quasi-Synchronous (Manivannan and Singhal,1999) algorithms.

Majority of the above approaches suffer from the overhead that result from forcing all the agents in multi-agent system to checkpoint. To overcome the problem of recovery latency and blocking, we propose coordinated checkpoint algorithm that is able to force the most limited number of agents carrying out process, for putting checkpoint. The concept of antecedence graphs (Khokhar et al, 2006) for fault tolerance in distributed systems was originally introduced in Manetho (Elnozahy,1993) which utilized antecedence graphs and message for mechanism for fault tolerance in distributed systems. But the overhead due to size of antecedence graph with large number of agents involved may cause greater overheads in case of multi-agent systems. Our proposed scheme significantly resolves the associated problem of overhead combining antecedence graph approach with non-blocking

Singh R., Kaur R. and Krishna Challa R. (2010).
ANTECEDENCE GRAPH APPROACH TO CHECKPOINTING FOR FAULT TOLERANCE IN MULTI AGENT SYSTEM.
In *Proceedings of the 12th International Conference on Enterprise Information Systems - Software Agents and Internet Computing*, pages 139-142
DOI: 10.5220/0002898601390142

checkpointing done by coordinating the time of checkpointing.

The rest of the paper is organized as follows: Section 2 describes the basic framework of the proposed scheme. Section 3 illustrates the algorithm of proposed scheme of checkpointing and recovery. Finally in Section 4, we give the performance analysis and results of comparison with existing schemes followed by conclusion about the effectiveness of proposed scheme in Section 5.

2 SYSTEM FRAMEWORK

The system consists of cooperating multiple agents (on a single or multiple mobile hosts) which form MA group and collaborate with each other to perform a single computationally complex task by passing messages between each other as shown in figure 1.

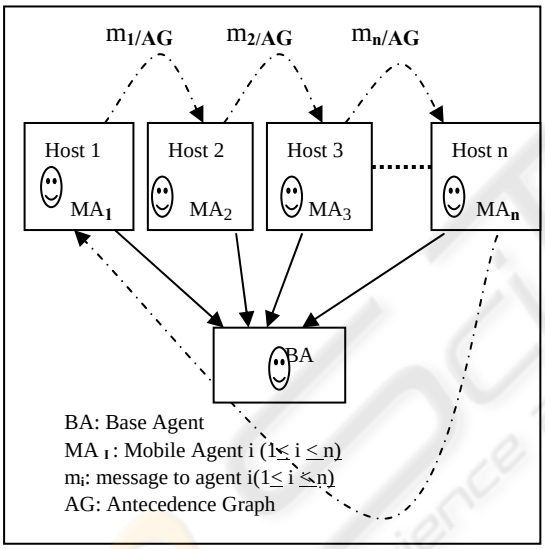

Figure 1: Multi Agent Group.

Each group has a Base Agent (BA) which coordinates the participating agents of group and is assumed to execute in fail safe mode. It also acts as recovery manager and maintains access to persistent data storage, where agent checkpoints and recovery bookkeeping is held. Under our strategy, each mobile agent will send its current antecedence graph to the agent that it is sending a message to. The mobile agents may perform checkpointing of the antecedence graph either when the depth exceeds certain threshold or after elapsing of specific time. The three basic steps involved in the proposed scheme are Formation of Antecedence graph at individual agents, Parallel checkpointing and Recovery in case of failure. These are discussed in detail in the following sections. We assume that all the operations executed by the mobile agents are idempotent, so the exactly once execution property needs not to be considered.

As an example, let us consider a scenario of a multi-agent system as shown in figure 2. For simplicity, we are only discussing three agents, agent A, agent B and agent C. Each agent, at the start of its execution, is at state Ω^0_A, Ω^0_B and Ω^0_C respectively. Each message receipt forms a deterministic interval. For example, the receipt of message m_1 from B to C forms the deterministic interval and the antecedence graph of state interval Ω^1_B provides information about what happened before.

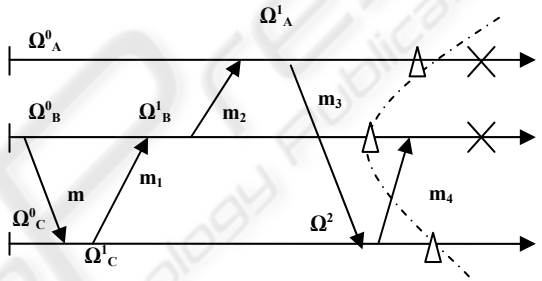

Figure 2: A multi-agent system with three agents.

3 PROPOSED CHECKPOINTING SCHEME

The main goal of proposed scheme is to minimize the global checkpointing latency and to reduce the total recovery time. In proposed scheme, the dependence information is accessible to the agent which requires for the checkpoint from its antecedence graph. When the antecedence graph depth exceeds certain threshold or after elapsing of certain time, an agent may request for checkpointing. For each MA_j, we set a variable Graph Depth (GD_j), which is the depth of requesting agent's antecedence graph at initialization of checkpointing. At threshold event, if MA_j starts a checkpoint request and informs all dependent agents (DA) of its antecedence graph. It carries out this request through a MA called CheckAgent (CA) which is made for every DA during the start of checkpoint agent and the time of sending checkpointing request to the DAs.

When MA_j sends this request, it attaches with CA, a numeric weight of value $1/|\ GD_j\ |$. In parallel

the requesting agent as well as dependent agents make a temporary antecedence graph of the events occurred during execution of checkpointing operation. The time of this temporary logging is overlapped with actual execution of the transaction and checkpointing and so it does not have any extra load for system and is therefore non blocking. The distinctiveness of our scheme is that the checkpoint request is distributed through all the agents in a parallel manner. After final checkpointing, the previous message logs and antecedence graphs are deleted which considerably reduces the size of the graph piggybacked on the message thereby helping to maintain the efficiency of algorithm in scenario where large number of agents participate in performing a transaction. After successful completion of checkpointing, the involved agents for construction of new antecedence graphs may continue from the temporarily saved antecedence graphs.

In case of failure the recovering agents request the BA to send the maximum length antecedence graph. The recovering agent reconstructs its own graph from the received last checkpointed antecedence graph. If in self state, MA_j decides for checkpointing, then would call following algorithm:

```
Requesting Agent MAj send for GDj  from
Dependent Agents(DA)
For each Agent ∈ Ancedence graph(AG)
          Create CheckAgent(CA)
MAj  send  a  CA  with  temp-checkpoint
request and value 1/|GDj | to all MAi (
where i ≤ j)
       W=0
   For each agent∈ AG
MAj receives reply to temp-check
request.
       for each reply compute:
        W=W + 1/|GDj|,
   if W≠1then
   cancel checkpointing & wait for
threshold event
   else if W=1 then
At MAj and all DAs:
    Save antecedence graph as
checkpoint.
    Send the final checkpointed AG to
BA.
    Discard suceessfully checkpointed
nodes from AG.
    Continue again from temporary AG.
At BA:
   Construct maximum length AG from
received AGs.
Write it to stable storage.
```

The checkpointed state at BA is used to provide fault tolerance and recovery in case of agent failure.

4 PERFORMANCE ANALYSIS AND COMPARTIVE STUDY

In proposed system multiple agents are performing in a group. Suppose that MA_k is related to MA_{k+1} in antecedence graph. In the scheme given in (Khokhar et al, 2006) as the checkpoint is not optimized the requesting agent sends the checkpointing request to other all the agents, if MA_k starts the checkpointing request, the checkpointing request distributes from MA_k to MA_1 through all the MA_{k-1}, MA_{k-2}, …, MA_2 and MA_1. In this case, the connection between the agent forms a message request path. So the length of this path is n-1 that is presented as Lkt(n). In the proposed scheme, in the most optimized form, there is one dependent agent for the agent that request the checkpoint and in the worst form, all the agents are dependent to this agent. This is the same n-1 that existed in the former scheme. So for this, the presented average is shown as:

$$Lc(n) = n/2$$
$$Lc(n)/Lkt(n) = \lim[\, n/2 / (n-1)] = 1 / 2$$

Due to space limitation, we are eliminating the detailed theoretical part.

To implement, we have used AGLETS (Lange 1998) that is a graphical interface for developing the distributed multi-agent systems. For the suggested scheme implementation, the tasks and the behaviour of every agent has been made in the form of classes. First for better verification and getting the more enhanced results, 170 agents are defined and made on the mobile host. Then the agents that manage these agents are activated in order to wait for the messages for the checkpointing. Each time some of these 170 agents are defined as the dependent agent and we measure the time of the checkpointing agent with the counter that has been provided in the graphical interface. We also test this environment using the scheme in (Khokhar et al, 2006). In this test a checkpoint message is sent to all the agents without regarding their dependency to the starting agent. Results as shown in figure 3 were obtained after the implementation of the checkpointing part of proposed scheme with a different list of the dependent agents out of these 170 agents. As it can be seen, as the number of the dependent agents is increased in relation to the total number of agents in group, the time increases and approaches to the scheme in (Khokhar et al, 2006).

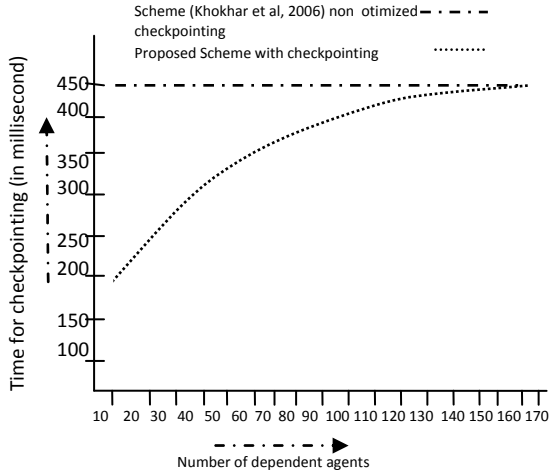

Figure 3: Comparison of time of checkpointing.

5 CONCLUSIONS

In this paper we proposed a strategy to introduce fault tolerance in multi agent system through checkpointing using antecedence graph approach. Our results show that checkpointing done through collection list of only dependent agents underlined by antecedence graphs could significantly improve the efficiency of checkpointing algorithms. Comparative analysis of our scheme with previous antecedence graph based schemes show reduction in recovery time and lower overheads.

REFERENCES

Nwana, Hyacinth, S., 1996. Software Agents: An Overview. Knowledge Engineering Review. Vol. 11, Cambridge University Press. pp. 1 – 40.

Lyu M. R., Chen, X., Wong. T. Y., 2004. Design and Evaluation of a Fault-Tolerant Mobile-Agent System. *IEEE CS Press*, pp. 32-38.

Elnozahy, E, Alvisi, N., L., Wang, Y, M., Johnson, D, B., 1999. Survey of Rollback-Recovery Protocols in Message- Passing Systems, Technical Report CMU-CS-99-148, School Computer Science, Carnegie Mellon University.

Khokhar, M, M., Nadeem, A., Paracha, O,M.,2006. An Antecedence Graph Approach for Fault Tolerance in a Multi-Agent System. *Proceedings of the IEEE 7th International Conference on Mobile Data Management.*

Elnozahy, E, N., 1993. Manetho: Fault Tolerance in Distributed Systems Using Rollback-Recovery and Process Replication, PhD Thesis, Rice University, Houston, Texas.

Meth, K, Z., Tuel, W, G., 2000. Parallel checkpoint/restart without message logging. *Proceeding of IEEE 28th Int. Conf. on Parallel Processing*, pp. 253-258.

Bhargava, B., Lian, S, R., 1998. Independent checkpointing and concurrent rollback for recovery in distributed systems - an optimistic approach, *Proceeding of 7th IEEE Symp. Reliable Distributed Syst*.,pp. 3-12.

Manivannan, D., Singhal, M., 1999. Quasi-synchronous checkpointing: Models, characterization, and classification, *IEEE Trans. Parallel and Distributed Syst.*, 10(7): pp.703-713.

Lange, B, Banny., 1998. Java Aglets Application Programming Interface(JAAPI) White Paper-Draft 2 , IBM Tokyo Research Laboratory.