A Checkpointing Scheme for the Mobile Application with a Long Lifetime

Sung Chae Lim

Dept. of Computer Science, Dongduk Women’s University
Seoul 136-714, Korea
sclim@dongduk.ac.kr

Abstract. When the mobile application has a long lifetime and it is comprised of multiple application processes running on more than one mobile device, it is required to make checkpoints for saving their intermediate processing states. In this paper, we present a checkpointing scheme suitable for the long-living distributed applications. For this, we deploy logging agents across mobile support stations so that they can efficiently collect the causality dependency vector to be used for recovery in the presence of hardware or software failure. Owing to the low network overhead and fast recovery time of our method, the proposed method can significantly reduce the overhead paid for synchronized checkpointing in a distributed mobile application.

Keywords: distributed checkpointing, recovery, mobile computing

1 Introduction

When the mobile application has a long lifetime and it is comprised of lots of parallel tasks, it is required to safely checkpoint its processing states against abrupt failure. To make the distributed application robust and recoverable against various types of failure, many earlier researches have been done for the computing environment where the distributed application is executed in the fixed networks [1, 4, 5, 6]. Although those previous algorithms are eligible for executing traditional distributed applications, it is hard to apply them toward the mobile computing environment because of many differences in the network architecture and used computer systems [2]. For instance, the mobile device, which runs AP’s in the case of the mobile computing environments, has tight limitations on the computing resources such as battery power, memory and disk storage. In particular, its communication cost through wireless networks is higher and unstable, compared with applications running in the fixed networks. For these reasons, some checkpointing schemes are proposed for the mobile computing environment as in [1, 2, 3].

In the paper, a new checkpointing scheme for distributed mobile applications is devised. For our scheme, we propose two key ideas of a checkpoint-related agent program and a flexible checkpointing protocol. The agent program is deployed at stable hosting servers, so called, mobile station site, and the CA creates and saves checkpoint records on behalf of AP’s for reducing the amount of data delivered via wireless networks. We call such an agent program the logging agent.
2 Basic Idea

The notion of the GCS (Global Consistent State) was formally defined in the research [4] by Lamport. Fig.1 shows an example where distributed checkpoint is performed by three AP’s, p1, p2, and p3. In the figure, the blacked rectangle of $C_{i,k}$ represents the $k$-th local checkpoint made by AP $p_i$. Suppose that an application failure arise at $p_3$. as in Fig. 1. At this moment, the set of local checkpoints preserving the GCS are that inside the GCS line of the figure. That is, the latest CGS state is composed of $C_{1,3}$, $C_{2,2}$, and $C_{3,3}$. As the message sending event of $m_8$ is not saved in any local checkpoint, its message receiving event cannot be include a GCS. Therefore, $C_{2,2}$ is rollbacked, and $C_{1,3}$ is also rollbacked because $C_{1,3}$ contains the message-receiving event of $m_7$ saved in $C_{2,2}$. As a result, the local checkpoints on the GCS line will be used to recover the failed application.

In the example of Fig. 1, the latest local checkpoints of a GCS are the same as $C_{1,3}$, $C_{2,2}$, and $C_{3,3}$ is used as a latest consistent global checkpoint. This choice of such a consistent global checkpoint is done by a recover algorithm initiated in the presence of failure.

3 Proposed Algorithm

Fig. 2 shows the way a CA works at the time when a network message $m$ arrives at the CA. In the algorithm, the CA receiving message $m$ is denoted by $C_i$, and the AP checkpointed by $C_i$ is denoted by $P_i$. The message $m$ can be one from $P_i$ or any other CA. Since all the messages sent to an AP are relayed by CA’s, every message from AP’s other than $P_i$ arrives at $C_i$ via the CA’s. For more detail, [1, 5] can be referred to.
**Used Data:** \( R \quad /* \text{current checkpoint record of } P_i */ \)

When a message \( m \) arrives at \( C_i \) from \( P_i 
1. \textbf{begin} \newline
2. \textbf{if} \ ( \ m \text{ is for requesting a checkpoint creation )} \textbf{then} \newline
3. \quad \text{Save the content of } R \text{ into disk space to make a local checkpoint with the serial number of } R\text{.serial.} \newline
4. \quad R' \leftarrow \text{GetGCSRec}(R). \quad /* \text{get a latest checkpoint record of a GCS */} \newline
5. \quad \textbf{if} \ ( \ R' = \text{nil} \ ) \newline
6. \quad \text{Call the routine } \text{GreateGCS}(R). \newline
7. \textbf{endif} \newline
8. \quad \text{Create a new checkpoint record with the serial number of } R\text{.serial} + 1. \newline
9. \quad \text{Send a messages notifying the creation of a new local checkpoint of } P_i. \newline
10. \quad \text{Send a response message of checkpointing to } P_i. \newline
11. \textbf{else} \quad /* \ m \text{ contains application data sent to other AP */} \newline
12. \quad \text{Append checkpoint-related fields to } m \text{ and send it to the counterpart CA.} \newline
13. \textbf{endif} \newline
14. \textbf{end.} \newline

Fig. 2. CA algorithm for handling a message-receiving event.

### 4 Conclusion

The problem of making a consistent distributed checkpoint in a mobile network environment is challenging to solve. This is because the distributed application needs application processes’ communication via wireless networks and such wireless communications easily make the cost for checkpoint higher. In this paper, we present a new checkpoint scheme using logging agent. Owing to the low network overhead and fast recovery time of our method, the proposed method is very suitable for checkpointing distributed mobile applications having long lifetime.

### References