VDAS: Variable Dependency After Send Family of Checkpoint Dependency Models in Distributed Systems

Chyi-Ren Dow  
Department of Information Engineering and Computer Science  
Feng Chia University, Taichung, Taiwan  
crdow@crab.ieecs.fcu.edu.tw

Cheng-Min Lin  
The Department of Electrical Engineering  
Nankai College, Nantou, Taiwan  
lcm@bear.nkjc.edu.tw

Abstract

Despite its simplicity and low run-time overhead, independent checkpointing may incur a high recovery overhead due to the domino effect caused by useless checkpoints. Although various adaptive checkpointing techniques have been developed to eliminate the domino effect, reducing the number of additional checkpoints that must be taken to prevent other checkpoints from becoming useless is of priority concern. Therefore, this investigation presents novel VDI and VDAS models which are generalizations of various domino effect-free models. Two efficient adaptive independent checkpointing algorithms are also proposed. The algorithms use \( n \) bits as piggyback information for a distributed system of \( n \) nodes. In addition, a visual simulator is developed and implemented to evaluate the effectiveness of various checkpointing models. Experimental results demonstrate that the proposed models are domino-free and the number of checkpoints taken is less than that of other models.

Keywords: Distributed systems, fault tolerance, checkpointing, rollback recovery, and checkpoint dependency models.

1 Introduction

Checkpointing is extensively used in many distributed/parallel applications, such as fault-tolerance [4, 11], debugging [3, 6], and mobile computing [9]. During normal execution, the state of each process is periodically saved on stable storage as a checkpoint. When a failure occurs, each process can then roll back to its previous checkpoint by reloading the saved state. A global checkpoint consists of a set of local checkpoints, each from different processes. A global checkpoint is consistent if and only if no local checkpoint arises before another (i.e. no messages are sent after a local checkpoint and received before another one).

In a message passing system, checkpointing consists of two approaches: coordinated [9, 13] and independent [1, 16]. The former involves a scenario in which a process coordinates other processes to save their state simultaneously. Therefore, the coordinator must send additional messages for synchronization. The latter does not require system-wide coordination and therefore may scale better [2]. Each process saves its state independently and does not require any additional message other than normal ones. Although lower than failure-free overhead of coordinated checkpointing, failure-free overhead of independent checkpointing may incur a higher recovery overhead due to the domino effect caused by useless checkpoints. The worst case scenario of the domino effect would be that all nodes restart execution.

Adaptive independent checkpointing and piggyback techniques can be used to eliminate useless checkpoints. For instance, if process \( P_i \) sends a message \( m \) with piggyback information to \( P_j \), \( P_i \) must take an additional checkpoint, referred to as adaptive checkpoint, before delivering the message \( m \) when it identifies a useless checkpoint in \( P_j \) based on the piggyback information. Although various adaptive independent checkpointing techniques have been developed to eliminate useless checkpoints [2], not all algorithms can ensure that all checkpoints are useful. For instance, the algorithm proposed by Xu and Netzer [16] may create useless checkpoints. Therefore, reducing the number of additional checkpoints that must be taken to prevent some other checkpoints from becoming useless is the primary task of adaptive checkpointing.

Model-based checkpointing is a form of communication-induced checkpointing [5] that maintains certain checkpoint and communication structure and can avert the domino effect. This work presents two novel models, the Variable-Dependency-Interval (VDI) and Variable-Dependency-After-Send (VDAS) models capable of providing a uniform view for various model-based checkpointing techniques. The proposed models are more general and have a lower failure-free overhead than other models. In addition, two adaptive independent checkpointing algorithms are proposed for the VDI and VDAS models.

The rest of this paper is organized as follows. Section 2 introduces the system model and various notations. Section 3 reviews pertinent literature.
Section 4 introduces the Checkpoint Dependency Graph (CDG). Section 5 then presents the VDI and VDAS models. In addition, two adaptive algorithms corresponding to the new models are also presented. Section 6 summarizes experimental results and, finally, conclusions are made in Section 7.

2 Preliminaries

A distributed system, represented by graph $G=(V, E)$, can be considered as a set $V$ of $n$ fail-stop nodes $P_i$ for $0 \leq i \leq n-1$, interconnected by a communication network consisting of a set $E$ of $m$ communication channels. There exists no shared memory or common clock among these nodes and the communication between the nodes is by message passing only. The channels are assumed herein to be FIFO order and reliable. Therefore, no messages will be lost but it may require finite amount of time to deliver.

The execution of a program in distributed systems can be modeled with $P = \langle E, \rightarrow \rangle$, where $E$ is a finite set of events and $\rightarrow$ is the happened before relation defined over $E$ [10]. The events include the execution of the following instructions, such as send, receive, or checkpoint operations. The $i$th checkpoint of process $P_i$ is denoted as $C_{i,i}$ and $C_{i,i-1}$ is assumed to be the initial checkpoint of process $P_i$. The time between two consecutive checkpoints, $C_{i,i}$ and $C_{i,i-1}$, is called checkpoint interval $I_{i,i}$, (including $C_{i,i}$ but not $C_{i,i-1}$).

3 Model-based Checkpointing

Communication-induced checkpointing [5] can be divided into two types: model-based checkpointing [12] and index-based coordination [7]. The former is domino effect-free and maintains certain checkpoint and communication structure. The latter enforces the consistency between checkpoints with the same index. In this section, we discuss several domino effect-free models for independent checkpointing, such as the CASBR, CBR, CAS, NRAS, FDI, and FDAS models [12].

For the CASBR model, the process must take a checkpoint after sending a message or before delivering a message. Hence, the number of checkpoints taken is twice the number of messages. An example of the model has been used in debugging for causal distributed breakpoints [3].

For the CAS model, the process takes a checkpoint only after sending a message. Hence, the number of checkpoints equals the number of messages. The model can also be used in the distributed breakpoints [6]. In addition, the model has been used to eliminate the problem of rollback propagation [14-15].

Although the CBR model resembles the CAS model, the process must take a checkpoint before delivering a message. The piecewise deterministic (PVD) model [11] can be viewed as an example of the CBR model. In the model, the execution of each process is divided into a deterministic state interval bounded by message receiving events or other nondeterministic events. The process must take a checkpoint before these events occur.

For the NRAS model, no receiving events occur after sending events. Hence, the process must take a checkpoint before delivering a message if at least one message is sent after the last checkpoint. Russell in 1980 proposed the MRS model [10] where M denotes Mark (or checkpoint), R represents Receive, and S is Send. The model can be viewed as an example of the NRAS model.

Although no useless checkpoint is created using the above four models, they may create many adaptive checkpoints, particularly when the message complexity increases. Hence, Wang [12] in 1997 presented FDI and FDAS models that combine checkpoint model and piggyback techniques to prevent useless checkpoints and to reduce the number of checkpoints for improving the run-time overhead. The Fixed-Dependency-Interval (FDI) model refers to a situation in which process $P_i$ is forced to take a checkpoint before processing any message that is about to change the transitive dependency vector $D_i$.

If a Boolean variable of sending messages is added into the FDI model then it changes into the FDAS model.

4 Checkpoint Dependency Graph

This section presents a novel notation, called the Checkpoint Dependency Graph (CDG). The CDG can be represented by $G = \langle C, E \rangle$ that consists of a set of nodes $C$, and a set of edge $E$. The nodes $C$ can be classified into a number of classes: volatile $C_v$, periodic $C_p$, adaptive $C_{ap}$, and useless $C_u$. A periodic checkpoint is taken periodically after a fixed time interval. An adaptive checkpoint is taken when a useless checkpoint is detected. A checkpoint is useless if the checkpoint is not a member of any consistent global cut. An edge of the CDG implies that a dependency relation arises between two checkpoints.

Figure 1 reveals that a time-space diagram can be translated into a CDG. Initially, initial checkpoints are taken before execution. Next, processes $P_1$ and $P_2$ send messages $m_1$ and $m_2$ to $P_3$ and $P_4$, respectively. Process $P_3$ takes a periodic checkpoint and then sends a message $m_3$ to process $P_5$. Finally, the time-space diagram is translated into the CDG as shown in Fig. 1a. This figure also contains a useless checkpoint $C_{0,1}$ because a zigzag cycle exists in the CDG. According to Fig. 1b, an adaptive checkpoint $C_{0,1}$ is taken before delivering the message $m_1$ by process $P_1$, implying that $C_{0,1}$ changes to a useful checkpoint after the adaptive checkpoint $C_{2,1}$ is taken.
5 VDAS Family of Checkpoint Dependency Models

This section presents two novel models: the Variable-Dependency-Interval (VDI) model and the Variable-Dependency-After-Send (VDAS) model. In addition to being more general than the FDAS models, these models can improve the failure-free overhead of the FDI and FDAS models, respectively.

5.1 Useful Checkpoint Models

As mentioned earlier, although the FDI and FDAS models can prevent the domino effect, many adaptive checkpoints can be taken for these models. Hence, two novel models, VDI and VDAS, are presented to enhance the performance of the FDI and FDAS models. Figure 2 illustrates a family of checkpoint and communication models. Notably, the VDI and VDAS models are generalizations of previous models such as the CASBR, CAS, CBR, NRAS, FDI, and FDAS models.

For independent checkpointing, many useless checkpoints may exist if it does not use any preventive technique of useless checkpoints. To resolve this problem, the eight domino effect-free models in Fig. 2 have been developed so far. These models are quite appropriate for service-providing applications. For instance, the VDAS model is used for computation oriented tasks to reduce the checkpoint overhead. Hence, the model can be applied for software error recovery, guaranteed deadlock recovery, and mobile computing. In contrast, the CASBR model is selected to debug a distributed program in order to obtain an earliest consistent global state. For instance, the model can be applied to causal distributed breakpoints. The models located at a higher portion of Fig. 2 yield a better performance than the models at a lower portion.

Definition 1. An independent checkpoint model belongs to the useful checkpoint model if it does not create any useless checkpoint.

5.2 Variable-Dependency-After-Send-Model

This subsection presents a novel model, VDAS, to extend the FDAS model. For the FDAS model, the process takes a checkpoint before processing any message that is about to change the transitive dependency vector after at least a message is sent. Herein, we release this constraint to allow for the change of transitive dependency vector as long as no useless checkpoints occur.

The performance of the VDAS model exceeds that of the FDAS model because it always takes fewer checkpoints. The FDAS model uses the transitive dependency vector to determine whether or not an adaptive checkpoint must be taken after sending at least a message. The vector consists of a set of checkpoint indices where each index belongs to a different process. Notably, a new adaptive checkpoint is taken when checkpoint dependency vector is changed. In contrast, the VDAS model allows for the change of transitive dependency vector and an adaptive checkpoint is taken if a useless checkpoint is identified. In the next section, we address an adaptive algorithm for the VDAS model.

Next, in Section 6, our experimental results demonstrate that the VDAS model exceeds the FDAS model in term of performance based on the experimental results. Figure 3 shows the difference between both models. According to Fig. 3a, checkpoints $C_{1,1}$ and $C_{2,2}$ are adaptive checkpoints which are taken when the checkpoint dependency vector is changed. In contrast, only one adaptive checkpoint exists in Fig. 3b. Notably, the checkpoint

![Fig. 1. The translation of time-space diagram to the CDG. (I or O: Periodic checkpoint, I or : Adaptive checkpoint, : Useless checkpoint)](image)

![Fig. 2. The VDAS family of checkpoint dependency models.](image)
$C_{2,i}$ is taken when $C_{1,i}$ is identified as a useless checkpoint.

As just mentioned, the VDAS model does not cause any useless checkpoint. Next, we present an efficient adaptive independent checkpointing algorithm to support the VDAS model for distributed systems. The algorithm uses a Boolean vector to determine whether or not to take an adaptive checkpoint in order to eliminate useless checkpoints. Every bit of this Boolean vector corresponds to a node in the distributed system. Process $P_i$ upon taking a new checkpoint and received at least one message in the previous checkpoint’s interval, establishes the values of all bits of this Boolean vector to true except bit $i$. Then $P_i$ sends the computational message in a piggybacked form with this vector to other nodes. This control information is used to inform other nodes that a new adaptive checkpoint may be necessary to prevent useless checkpoints. Each node maintains the following data structure:

**maybe_useless**: a vector of $n$ Boolean values at each node. The vector $maybe_useless$ records whether or not a useless checkpoint may occur in process $P_i$. Process $P_i$ upon sending a message to process $P_j$, piggybacks this information with the normal computation message to inform process $P_j$ whether an adaptive checkpoint must be taken to prevent useless checkpoints in process $P_j$. The vector is initialized to false.

**has_delivery**: a Boolean flag at each process $has_delivery$ denotes that at least a message has been delivered at process $P_i$. The initial value of this flag is false.

**send_to**: a Boolean variable is used to record whether a message has been sent after the last checkpoint has been taken.

Figure 4 formally describes the algorithm. The algorithm contains four major phases: the initialization phase, message-sending phase, message-receiving phase, and checkpointing phase.

Figure 5 illustrates an adaptive independent checkpointing example under the VDAS model. Each process $P_i$ is assumed to take a checkpoint $C_{1,0}$ initially and the Boolean vector $maybe_useless$ is set

```plaintext
var maybe_useless[1..n]; boolean;
send_to, has_delivery; boolean;

(1) Actions for the initialization by $P_i$
   ∀k do maybe_useless[k] := false enddo;
take_checkpoint;

(2) Actions taken when $P_i$ sends a message to $P_j$
   send_to := true;
   send($P_j$, message, maybe_useless) to $P_j$;
   maybe_useless[i] := false;

(3) Actions for node $P_i$ when receiving a message from $P_j$
   receive($P_j$, message, maybe_useless) from $P_j$;
   if send_to = true and maybe_useless[i] = true then take_checkpoint; endif;
   has_delivery := true;
   ∀k, k ≠ i do maybe_useless[k] := maybe_useless[k] OR maybe_useless[k] enddo;
   deliver(message);

(4) Action for $P_i$ to take a checkpoint
   take_checkpoint;

(5) procedure take_checkpoint is
   if has_delivery = true then ∀k ≠ i do maybe_useless[k] := true enddo; endif;
   send_to := false;
   has_delivery := false;
```

Fig. 4. An adaptive checkpointing algorithm for the VDAS model.
Next, process $P_i$ must set all bits of the $\text{maybe\_useless}$ flag to true except itself after taking a periodic checkpoint $C_{0,1}$. This is attributed to the fact that process $P_i$ takes this checkpoint after receiving message $m_j$ from process $P_j$. Then, process $P_i$ sends messages $m_j$ and $m_i$ to process $P_j$, and the piggybacked vectors \{F, T, T\} and \{F, F, T\} are appended after them, respectively. Process $P_j$ takes an adaptive checkpoint only before delivering message $m_j$ because no message is sent after checkpoint $C_{1,1}$. Finally, process $P_j$ must take an adaptive checkpoint to prevent a zigzag cycle that consists of $m_j$, $m_i$, and $m_j$.

### 6 Experimental Results

In this section, numeric results are collected, with these results demonstrating that the performance of our algorithm exceeds that of other algorithms. Our simulation model is used for a system with $N_p$ nodes connected through a general network. In the proposed model, several parameters use the Poisson distribution. Each process initially sends out messages with a rate $\lambda_i$, following exponential distribution and, then, the receiving sites of these messages are randomly selected. According to $\lambda_i$, when a new message arrives in the sender, a random function is performed for generating the destination of the message. The second parameter is checkpoint rate $\lambda_c$ so that the mean checkpoint interval time is $1/\lambda_c$. The parameter is limited for taking periodic checkpoints but not for adaptive checkpoints. $L_m$ is the latency of message transmission.

For simplicity, assume the following conditions for the simulation. Initially, assume that the processes stop the execution at the same time. Furthermore, the total execution time of the simulated program does not include the checkpoint time, denoted as $T_{	ext{RUP}}$. The second assumption is that no failure events occur to obtain the number of useless checkpoints. Next, checkpoint saving time is also assumed to be one computational unit. Finally, assume that each process takes an initial checkpoint before execution. The checkpoint provides a basic recovery point when a failure occurs.

For independent checkpointing, most of the checkpoints are useless particularly when the number of processes increases as shown in Fig. 7. For instance 60% of the checkpoints are useless when the number of processes is greater than fourteen. Furthermore, roughly 20% of checkpoints are also useless for two processes. In sum, this experimental results indicate that independent checkpointing causes the domino effect because most of checkpoints are useless. Hence, the checkpoint model must be used to prevent useless checkpoints.

---

**Fig. 5.** An adaptive checkpointing example under the VDI model.

**Fig. 6.** An adaptive checkpointing example under the VDAS model.

**Fig. 7.** Number of useless checkpoints for independent checkpointing.
According to Fig. 8, the VDAS model takes the lowest number of checkpoints among various models except the conventional independent checkpointing (IC) model. In general, the run-time overhead increases with an increase of the number of checkpoints. On the other hand, less adaptive checkpoints are expected to improve the failure-free run-time overhead. Fig. 8 illustrates that the VDAS model is better than other models. The number of checkpoints taken for adaptive algorithms in Fig. 8 (except the VDAS model) is nearly twice that of the number of IC model.

For the number of adaptive checkpoints, our algorithms are superior to others. Less adaptive checkpoints taken for independent checkpointing, the better performance of the algorithm is obtained since the overhead of checkpointing is smaller. Figure 9 indicates that although the rate of adaptive checkpoints for the VDAS model is nearly 60% but others are nearly 80%. Hence, the number of adaptive checkpoints taken for the VDAS model is smallest.

Figure 10 illustrates the checkpoint overhead vs. the number of processes. The overhead of all models, except the IC and VDAS models, increases with an increase of the number of processes. In addition, the checkpoint overhead of the IC is smaller than that of other models because it only takes periodic checkpoints; however, no adaptive checkpoints are taken. However, it can incur the domino effect. Hence, the VDAS model is best with respect to checkpoint overhead as shown in Fig. 10.

In addition to the conventional independent checkpointing, the number of checkpoints decreases with an increase of the message interval. Notably, the number of checkpoints approaches to that of the conventional independent checkpointing if the message interval exceeds the checkpoint interval. Figure 11 indicates that the VDAS model is better than other models in terms of various message intervals.

For different checkpoint intervals, Fig. 12 demonstrates that the VDAS model except the conventional independent checkpoint is better than other models. In addition, the number of checkpoints decreases with a decrease of the checkpoint interval increases.
7 Conclusions

Despite its lower failure-free overhead, independent checkpointing has a higher recovery overhead caused by the domino effect. Although various checkpoint models of communication-induced checkpointing have been developed to avert the domino effect for independent checkpointing, these models have a common limitation that many adaptive checkpoints may need to be taken. Hence, these models incur a higher failure-free overhead. In this work, we present two new models. The proposed models not only eliminate the domino effect, but also improve the failure-free overhead in checkpointing. Furthermore, the VDAS family can provide a uniform view for model-based communication-induced checkpointing.

This investigation also presents two adaptive independent checkpointing algorithms which corresponding to the two models proposed herein. The two algorithms use only a Boolean vector consisting of n bits as piggyback information. The piggyback information is superior to that of other proposed models such as FDI and FDAS. Results presented herein demonstrate not only the validity of our methods, but also the effectiveness of our algorithms. Furthermore, experimental results indicate that the two algorithms do not generate any useless checkpoint and the number of checkpoints taken is less than that of other models.

In sum, this paper presents two novel models capable of providing a uniform view for various model-based checkpointing techniques. The proposed models also have a lower failure-free overhead than the FDAS models. Contributions of this paper are

1) Two new models are proposed: the VDI and VDAS models. The proposed models are more general than the family of FDAS models. In addition, the models are domino effect-free and yield a better performance in
run-time overhead than in previous models.
2) Two adaptive independent checkpointing algorithms are proposed for the VDI and VDAS models. The two algorithms use only a vector that consists of $n$ bits as piggyback information for a distributed system of $n$ nodes.

References