결함 내성 분산 시스템에서의 동적 검사점 스케줄링 기법
(A Dynamic Checkpoint Scheduling Scheme for Fault Tolerant Distributed Computing Systems)

박 태 순 †
(Taesoon Park)

요 약 분산 시스템에 결함 내성 기능을 제공하는 기법의 하나인, 검사점을 이용한 회복 기법을 효율적으로 구현하기 위해서는 최적화 된 검사점 설정 구간의 선택이 매우 중요한 문제로 인식되고 있다. 본 논문은 분산 시스템내의 각 프로세스가 적절한 검사점 설정 구간을 프로세스의 연산 중에 동적으로 스케줄링 하는 기법을 제안한다. 제안된 기법에서는 시스템내의 각 프로세스가 현 검사점 구간 동안의 검사점 설정 비용과 가능한 롤백 회복 비용을 비교 평가하고, 다음 검사점 설정을 위한 적절한 구간을 계산한다. 대부분의 기존 기법들과는 달리, 제안된 기법은 검사점과 롤백 두 가지 비용 모두를 최소화하는 구간 값을 선택하며, 현 검사점 구간 동안의 통신 형태를 고려한 구간 값을 선택한다. 또한, 검사점 설정 구간 선택을 위한 별도의 통신비용이 요구되지 않으며, 제안된 기법은 기존의 검사점 조정 기법들과 쉽게 통합되어 사용될 수 있다.

키워드 : 결함 내성 시스템, 분산 시스템, 검사전, 롤백 복구, 동적 스케줄링

Abstract The selection of the optimal checkpointing interval has been a very critical issue to implement a checkpointing-recovery scheme for the fault tolerant distributed system. This paper presents a new scheme that allows a process to select the proper checkpointing interval dynamically. A process in the system evaluates the cost of checkpointing and possible rollback for each checkpointing interval and selects the proper time interval for the next checkpointing. Unlike the other schemes, the overhead incurred by both of the checkpointing and rollback activities are considered for the cost evaluation, and the current communication pattern is reflected in the selection of the checkpointing interval. Moreover, the proposed scheme requires no extra message communication for the checkpointing interval selection and can easily be incorporated into the existing checkpointing coordination schemes.

Key words : Fault-Tolerant System, Distributed System, Checkpointing, Rollbacdk-Recovery, Dynamic Scheduling

1. Introduction

Checkpointing is an operation to save correct intermediate states of a process into a stable storage which is not affected by system failures. With the periodic checkpointing, a process can resume the computation from one of the saved states when it recovers from a system failure. The saved state of a process is called a *checkpoint* and the action to resume the computation from a checkpoint is called a *rollback*. Checkpointing is an effective tool to prevent a process from restarting its computation from the initial state even in case of a system failure. However, frequent checkpointing severely degrades the system performance since it takes a non-negligible amount of time to save a checkpoint. Hence, the selection of the optimal checkpointing interval has been a critical issue to implement the checkpointing, and many

† 종신회원 : 세종대학교 컴퓨터공학과 교수
tspark@kunja.sejong.ac.kr
논문접수 : 2001년 3월 21일
심사완료 : 2001년 10월 10일

analytic models have been suggested to compute the optimal checkpointing interval for a single processor system in which no message communication among the processes is considered [1,2,3].

In a distributed system, the states of the processes running in the system become dependent on one another due to the message communication. The state S_i of a process P_i is said to be *directly dependent on* the state S_j of another process P_j; (1) if at the state S_j, the event of sending a message m happens and S_i is the state caused by the event of receiving m; or (2) if $P_i = P_j$ and S_i immediately follows S_j. Also, the state S_i of P_i is said to be *transitively dependent on* the state S_j of P_j and the relation is denoted by $S_j \rightarrow S_i$, if the relation between the two states, S_i and S_j, is found in the transitive closure of the "directly dependent on" relation [4]. Because of this inter-process dependency, the failure of one system site may force the rollback of not only the processes running on that site but also the ones running on the other sites.

For example, suppose that a process P_i should roll back to a checkpoint C^i after a failure. Then, the new computation performed from C^i may not be the same as the one which has been performed before the rollback, since the non-deterministic computation can be used or the messages received by P_i before the rollback may not be retrieved in the same order as before. In such a case, the processes carrying the states dependent on P_i's old computation invalidated by the rollback also have to roll back to the checkpoints taken before the dependency relation was formed. In other words, for a dependency relation, $S_j \rightarrow S_i$, if the state S_j is lost due to a failure, the state S_i also has to be discarded by the rollback. It is said that the processes roll back to a *consistent recovery line*, if no state dependent on the invalidated states of the recovering process remains in the system after the rollback-recovery [5].

When the rollback actions are propagated to the related processes, the processes may have to be involved in a *domino effect* [6], which causes the processes to roll back recursively to reach the consistent recovery line and in the worst case, enforces the processes to roll back to their initial states. Figure 1 shows an example of the domino effect. In the figure, the process P_i first rolls back to its latest checkpoint $C^{i,2}$ after a failure. Then, due to the dependency caused by *m5*, P_j also has to roll back to the checkpoint $C^{j,2}$, and the rollback of P_j causes the second rollback of P_i to the checkpoint $C^{i,1}$ which has been taken before the receipt of *m4*. In a similar way, the dependency relations caused by *m3*, *m2*, *m1* force both of P_i and P_j to roll back to their initial states.

The main reason to cause the domino effect is the *backward dependency* [6], in which an old checkpointing interval of a process becomes dependent on a recent checkpointing interval through the abnormal dependency among the checkpointing intervals as shown in Figure 1. Let $I^{i,\alpha}$ denote the state interval between two checkpoints $C^{i,\alpha-1}$ and $C^{i,\alpha}$, where $\alpha > 0$, and the relation $I^{j,\beta} \rightarrow I^{i,\alpha}$ indicates that for any $S_j \in I^{j,\beta}$ and $S_i \in I^{i,\alpha}$, $S_j \rightarrow S_i$. Then, the backward dependency means that for any two intervals, $I^{i,\alpha-k}$ and $I^{i,\alpha}$, $I^{i,\alpha-k} \rightarrow I^{i,\alpha}$ and also $I^{i,\alpha} \rightarrow I^{i,\alpha-k}$, where k is a positive integer. For example, in Figure 1, it is natural that $I^{i,2} \rightarrow I^{i,3}$. However, due to the message *m5* and *m4*, the relations, $I^{i,3} \rightarrow I^{j,3}$ and $I^{j,3} \rightarrow I^{i,2}$, hold, and as a result, the backward dependency, $I^{i,3} \dashrightarrow I^{i,2}$ also holds.

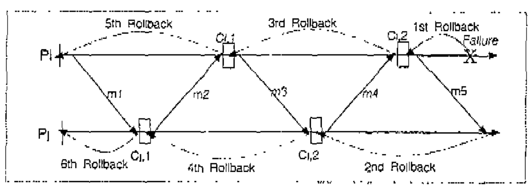

Fig. 1 A Domino Effect

One way to avoid the domino effect is the *careful coordination of checkpointing* actions of the related processes and various checkpointing coordination schemes have been suggested [5,7,8, 9,10,11,12]. In these schemes, a process takes the checkpoints on its own schedule, which is called the *scheduled checkpointing,* and also takes the checkpoints on the coordination request from another process, which is called the *forced checkpointing*. Hence, to compute the optimal checkpointing interval for the processes running in a distributed system, the effects of forced checkpointing and rollback propagation have to be considered, which makes the selection of the optimal checkpointing interval for the communicating processes enormously difficult, and little research efforts have been made.

Instead of selecting the optimal interval, the schemes in the literature adjust the checkpointing interval by varying the conditions to take forced checkpoints so that proper performance can be achieved. A scheme proposed in [11] achieves the minimal computation loss in case of a system failure by taking forced checkpoints whenever a new dependency relation is formed. Instead, the number of forced checkpoints in this scheme can be excessive. The schemes proposed in [7,8] take the less number of forced checkpoints by assigning a sequence number to each checkpointing interval and taking the forced checkpoints at the computational points to avoid the backward dependency. The schemes in [12] further reduces the number of forced checkpoints by partially allowing the domino effect. However, as it is noticed from the previous experimental results, the schemes pursuing the less number of forced checkpoints have to experience a large amount of computation loss after rollback; and the ones reducing the rollback and recovery cost have to endure frequent checkpointing [12].

In this paper, a new scheme to adjust the checkpointing interval considering the forced checkpointing and the rollback propagation is presented for a distributed system. In the proposed scheme, a process selects the time interval for its scheduled checkpointing considering the forced checkpointing frequency, the expected computation loss, and the expected number of rollbacks. Unlike the solutions for the single-processor system, the checkpointing interval in the proposed scheme is dynamically adjusted to reflect the dynamic changes of the inter-process dependency. With the proper adjustment of the effects of the forced checkpointing and the rollback, the performance of the processes can significantly be improved. Moreover, such adjustment can be performed at each process based on its local information without any extra communication, and the proposed scheme can easily be incorporated into any of the existing coordination schemes.

2. System Model

A distributed system consisting of a number of nodes connected through a communication network is considered. Logically, the system can be viewed as a set of processes running on the nodes and a set of communication channels between every pair of the processes. A process is considered as a sequence of state transitions from the initial state to the final state. An event is an atomic action that causes the state transition within a process, and a sequence of events is called a *computation*. The processes are not assumed to share the memory, but to communicate only through message passing. The communication subsystem is assumed to be reliable; that is, the message delivery is error-free and virtually lossless. A process is also the unit of failure and recovery in the system. When a node fails, every process running on the node is declared to fail. A fail-stop node is assumed [13] and only the transient failures are considered in the system; that is, once a process recovers from a failure and re-executes the computation, the same failure is not likely to occur again.

3. Dynamic Checkpoint Scheduling

3.1 Checkpointing Scheme

Each process in the system takes scheduled checkpoints according to its own schedule and in addition, it takes forced checkpoints on the coordination request from another process. For the checkpointing coordination, any of the existing schemes can be used with the proposed checkpoint scheduling scheme. In this paper, the scheme proposed in [8] is considered and briefly introduced first. Let the *checkpointing interval* denote the state interval between two consecutive checkpointing of a process. Then, in [8], the checkpointing intervals of a process are sequentially numbered and each message sent out during an interval carries the interval sequence number. When a process receives a message with the attached sequence number higher than its current checkpointing interval number, it takes a new checkpoint before processing the message and assigns the number carried in the message to the new interval. As a result, no backward dependency, the dependency from a higher numbered interval to the lower numbered interval, can be formed and the domino effect is not possible.

Figure 2 shows an example of the checkpointing coordination scheme presented in [8]. The white rectangles in the figure denote the scheduled checkpoints; and the gray ones denote the forced checkpoints. The number shown inside each rectangle denotes the checkpoint sequence number and the one right aside the arrow denotes the sequence number carried in a message. As shown in the figure, each process takes a scheduled checkpoint on its own schedule and on the receipt of a message with the checkpoint sequence number larger than its current checkpoint sequence number, the process takes a forced checkpoint.

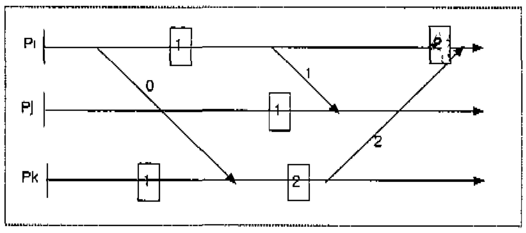

Fig. 2 An Example of Checkpointing Coordination

A process in the system can adjust the time interval for its scheduled checkpointing by properly setting the checkpointing timer. However, it is not easy to adjust the frequency of the forced checkpointing, since the forced checkpoints are taken on the receipt of a certain message from another process. Instead, the frequency of forced checkpointing of a process can be adjusted by another process, since a forced checkpoint of a process is taken following the scheduled checkpointing of another communicating process. Hence, for a given message communication rate, the number of forced checkpoints can also be determined by the scheduled checkpointing interval. To achieve the optimal performance of the system, the scheduled checkpointing interval must be selected toward reducing both of the checkpointing cost and the computation loss due to a rollback, called the *rollback distance*. Hence, the proposed scheme uses the dynamic adjustment of scheduled checkpointing interval to reflect the dynamic changes of inter-process dependency.

Let $I_s^{i,\alpha}$ denote the α-th scheduled checkpointing interval which is the sequence of events (or computational states generated by the events) between the completion of the (α-1)-th scheduled checkpointing and the completion of the α-th scheduled checkpointing of a process P_i; and $T_s^{i,\alpha}$ denote the time duration elapsed in $I_s^{i,\alpha}$. When each P_i completes the saving of the ($\alpha-1$)-th checkpoint, it schedules the time for the next scheduled checkpointing; that is, the expected value of $T_s^{i,\alpha}$ is determined. To select an appropriate value for $T_s^{i,\alpha}$, P_i should consider two factors: the checkpointing cost during the interval $I_s^{i,\alpha}$ and the possible failure-recovery cost during $I_s^{i,\alpha}$. The checkpointing cost for each $I_s^{i,\alpha}$ is the amount of time spent to take the scheduled and forced checkpoints during $I_s^{i,\alpha}$. Let $N_{fcp}^{i,\alpha}$ denote the number of forced checkpoints taken during $I_s^{i,\alpha}$ and T_{cp} be the expected time spent to save a checkpoint. Then, the checkpointing cost during $I_s^{i,\alpha}$, denoted by $C_{cp}^{i,\alpha}$, can be calculated as,

$$C_{cp}^{i,\alpha} = (N_{fcp}^{i,\alpha} + 1) * T_{cp}.$$

To obtain the value of $N_{fcp}^{i,\alpha}$, each P_i maintains a counter incremented by one at each time when a forced checkpoint is taken; and reset when a new scheduled checkpoint is taken. The value of T_{cp} can be obtained by measuring the time to save each of the checkpoints and then averaging them. Figure 3 describes the notations of $C_s^{i,a}$, $I_s^{i,a}$, $T_s^{i,a}$, and T_{cp}, in detail, in which $C_s^{i,a}$ and $I_s^{i,a}$ denote a checkpointing event and a sequence of events or computational states, respectively; while $T_s^{i,a}$ and T_{cp} denote the time spent for them.

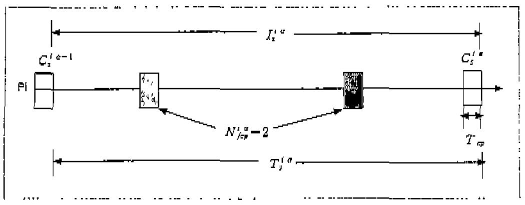

Fig. 3 An Example of a Checkpointing Interval

The possible failure-recovery cost for each $I_s^{i,a}$ is the expected time to restore the selected checkpoint and the expected amount of rollback distance with the probability of rollback happening during the interval. Let $N_{rb}^{i,a}$ denote the expected number of rollbacks during $I_s^{i,a}$ and $T_{rd}^{i,a}$ be the expected rollback distance in case that P_i has to roll back during $I_s^{i,a}$. To restore a selected checkpoint, the same amount of time to save a checkpoint, T_{cp}, is assumed. Then, the failure-recovery cost during $I_s^{i,a}$, denoted by $C_{fr}^{i,a}$, can be calculated as,

$$C_{fr}^{i,a} = N_{rb}^{i,a} * (T_{cp} + T_{rd}^{i,a}).$$

To calculate the value of $T_{rd}^{i,a}$, each process P_i measures the rollback distance for each rollback and obtains the normalized average rollback distance with respect to the scheduled checkpointing interval as follows. When P_i resumes the computation after saving or restoring a checkpoint, it saves its local clock value, say t_0, into the stable storage. When P_i has to rollback at time t_1, it calculates the rollback distance as $t_1 - t_0$. However, if P_i has to roll back due to its own failure, it cannot access its local clock to get the value of t_1. In such a case, either the average value of the previous rollback distance values or the value calculated from the simplified mathematical analysis can be used. Refer to the following subsection for the detailed description of the mathematical analysis.

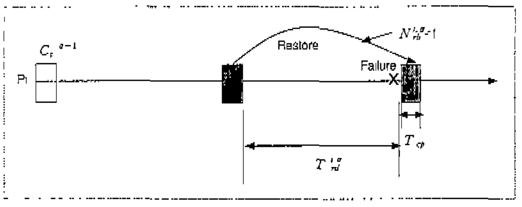

Fig. 4 An Example of Rollback Distance

Now, let $D_{rd}^{i,\beta}$ denote the rollback distance of the β-th rollback of P_i and $T_s^{i,\beta}$ denote the selected time interval between two scheduled checkpointing when the β-th rollback happens at P_i. Then, the mean ratio of the rollback distance to the scheduled checkpointing interval can be calculated as $\frac{1}{N_{rb}} * \sum_{\beta=1}^{N_{rb}} \frac{D_{rd}^{i,\beta}}{T_s^{i,\beta}}$, where N_{rb} is the number of rollbacks which have happened at P_i and the value of $T_{rd}^{i,a}$ is calculated as $\frac{1}{N_{rb}} * \sum_{\beta=1}^{N_{rb}} \frac{D_{rd}^{i,\beta}}{T_s^{i,\beta}} * T_s^{i,a}$. To calculate the value of $N_{rb}^{i,a}$, each process P_i maintains a counter, N_{rb}^{i}, to count the number of rollbacks which have been performed by P_i and also maintains a variable T_E^i to measure the time elapsed at P_i from the beginning of its execution. Then, $N_{rb}^{i,a}$ can be calculated as $\frac{N_{rb}^{i}}{T_E^{i}} * T_s^{i,a}$, which is the expected number of rollbacks during $I_s^{i,a}$ assuming the uniform distribution of rollbacks throughout the computation. Figure 4 describes the notations of $T_{rd}^{i,a}$ and $N_{rb}^{i,a}$, in detail, where P_i fails in $I_s^{i,a}$ and rolls back to its latest forced checkpoint.

Now, it is presented how the calculated values of $C_{cp}^{i,a}$ and $C_{fr}^{i,a}$ are used to adjust the scheduled checkpointing interval. Suppose that each process P_i in the system select a very large value for the scheduled checkpointing interval. Then, P_i may take the less number of scheduled checkpoints throughout the computation, and it may also force and be forced to take the small number of forced checkpoints. Hence, the checkpointing cost during the normal execution must be low. However, the failure-recovery cost of P_i can be high, since the expected rollback distance of P_i must be very large in case of a rollback. As a result, the overall performance of P_i can be degraded with a long checkpointing interval. Meanwhile, if each P_i in the system selects a very small value for the scheduled checkpointing interval, the rollback distance of P_i can be small and hence, the failure-recovery cost of P_i can be low. However, the overall performance of P_i may still be degraded due to the large number of scheduled and forced checkpointing.

To improve the overall performance of a process, the scheduled checkpointing interval must be adjusted to reduce both of the checkpointing and failure-recovery costs. Hence, in the proposed scheme, at the beginning of the execution, each process P_i arbitrary sets the time interval value for its first scheduled checkpointing. Then, at each time when P_i takes a new checkpoint, it evaluates the current checkpointing interval value based on the checkpointing cost and the possible failure-recovery cost to examine whether the current interval is too large or too small. For the next scheduled checkpointing interval, P_i uses the evaluation results to properly reduce or enlarge the time interval value as follows:

Case 1) Suppose that for the current scheduled checkpointing interval, $I_s^{i,a}$, the checkpointing cost, $C_{cp}^{i,a}$, is larger than the failure-recovery cost, $C_{fr}^{i,a}$: Then, P_i may have spent too much time to take the checkpoints compared to the possibly small effects of failure-recovery during $I_s^{i,a}$. Hence, P_i should try to reduce the number of checkpoints by increasing the time interval value for the next scheduled checkpointing. That is, $T_s^{i,a+1} = T_s^{i,a}*(1+K)$, where $K>0$.

Case 2) Suppose that for the current scheduled checkpointing interval, $I_s^{i,a}$, the checkpointing cost, $C_{cp}^{i,a}$, is smaller than the failure-recovery cost, $C_{fr}^{i,a}$: Then, P_i may have spent too little time for the checkpointing compared to the possibly large effects of failure-recovery during $I_s^{i,a}$. With the current value of $T_s^{i,a}$, the rollback distance of P_i can be too large in case of a rollback. Hence, P_i should reduce the rollback distance by decreasing the time interval value for the next scheduled checkpointing. That is, $T_s^{i,a+1} = T_s^{i,a}*(1+K)$, where $K<0$.

As a result, the time interval value for the scheduled checkpointing can be gradually adjusted to produce a better performance, as each process takes its scheduled checkpoints. The value of K can be another adjustable parameter. When the initial checkpointing time interval is considerably deviated from the optimal value, the process can rapidly reach the near optimal value using the coarse value of K. However, for the fine tuning to obtain the optimal value, the value of K must carefully be selected for the little adjustment of the checkpointing time interval.

3.2 Evaluation of Checkpointing Interval Using Mathematical Analysis

One possible problem of using the empirical data to calculate the failure-recovery cost is that the obtained value of $C_{fr}^{i,a}$ may not well reflect the actual rollback behavior of the system at the beginning of the process execution; that is, before enough rollbacks happen at each process. One way to solve this problem is to use a simple analysis to calculate the value of $N_{rb}^{i,a}$ and $T_{rd}^{i,a}$. For the simplicity of the analysis, the following assumptions are made:

(1) The inter-arrival time of failures of a process follows an exponential distribution with a rate λ_f.

(2) In case of a failure, instant recovery of the process is assumed.

(3) The time to determine a consistent recovery line is assumed to be negligible.

(4) The time interval between two consecutive message sending by a process follows an exponential distribution with a rate λ_s.

(5) The recipient of each message is randomly selected.

(6) The time to take the α-th checkpoint of the processes in the system is synchronized.

Because of the assumption (6), the processes in the system need not take any forced checkpoints. This assumption is made only for the simplicity of the failure-recovery cost analysis and the effects of the forced checkpoints on the values of $N_{rb}^{i,\alpha}$ and $T_{rd}^{i,\alpha}$ will be considered later. From the assumptions (2) and (3), when a process recovers from a failure, the time for the dependent processes to resume their computation after their rollback is also synchronized and the rollback points of the processes are always their latest checkpoints. Figure 5 shows an example of the rollback propagation under the assumption for a system consisting of four processes, P_1, P_2, P_3 and P_4.

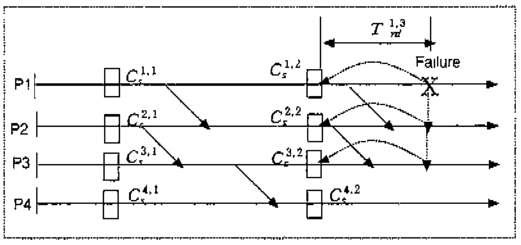

Fig. 5 Analysis Model

First, the value of $N_{rb}^{i,\alpha}$ is calculated. When a process recovers from its own failure or receives a rollback request from another process recovering from a failure, the process rolls back to its latest checkpoint. Let $N_f^{i,\alpha}$ be the expected number of failures in the system during $I_s^{i,\alpha}$ and $P_{rb}^{i,\alpha}$ denote the probability that a process is selected to roll back when a process fails during $I_s^{i,\alpha}$. Then, $N_{rb}^{i,\alpha}$ can be calculated as

$$N_{rb}^{i,\alpha} = N_f^{i,\alpha} * P_{rb}^{i,\alpha}.$$

As shown in Figure 5, when a process P_1 fails in $I_s^{1,3}$, processes P_2 and P_3 have to roll back with P_1 because of the dependency. However, P_4 does not have to roll back together, since it has no dependency on the lost state intervals of P_1. Since $P_{rb}^{i,\alpha}$ denotes the probability that a process has to roll back in case of a system failure, the value of $P_{rb}^{i,\alpha}$ becomes 0.75 in this example, assuming a uniform probability. More precisely, the value of $N_f^{i,\alpha}$, $P_{rb}^{i,\alpha}$, and $N_{rb}^{i,\alpha}$ can be calculated as follow.

From the assumption (1), the inter-arrival time of failures in the system follows an exponential distribution with a rate of $\lambda_f * N$, where N is the number of processes in the system. Hence,

$$N_f^{i,\alpha} = \lambda_f * N * (T_s^{i,\alpha} + T_{cp}).$$

When a process P_i recovers from a failure during $I_s^{i,\alpha}$, the processes whose current checkpointing interval is directly or transitively dependent on $I_s^{i,\alpha}$ must be selected to roll back together. A checkpointing interval $I_s^{i,\alpha}$ of a process P_i is said to be directly dependent on the checkpointing interval $I_s^{j,\beta}$ of another process P_j, if any state in $I_s^{i,\alpha}$ is directly dependent on a state in $I_s^{j,\beta}$. A checkpointing interval $I_s^{i,\alpha}$ of a process P_i is said to be transitively dependent on the checkpointing interval $I_s^{j,\beta}$ of another process P_j, if the relation between $I_s^{i,\alpha}$ and $I_s^{j,\beta}$ is found in the transitive closure of the ''directly dependent on'' relation [5].

Now, suppose that a process P_f fails during $I_s^{f,\delta}$ and rolls back to its latest checkpoint $C_s^{f,\delta-1}$. Let $T_{rd}^{f,\delta}$ denote the rollback distance of P_f which is the time interval between the latest checkpointing of P_f and the failure of P_f. Figure 5 shows an example of $T_{rd}^{1,3}$ when P_1fails during $I_s^{1,3}$. Then, $P_{rb}^{i,a}$ is basically the probability that a dependency relation from $I_s^{f,\delta}$ to $I_s^{i,a}$ is formed during $T_{rd}^{f,\delta}$. Let E_k be the set of events that there exists a transitive dependency relation consisting of k direct dependency relation from $I_s^{f,\delta}$ to $I_s^{i,a}$, where $k=0,1,2,\ldots,N-1$. Under our analysis model, only the most recent checkpointing interval of a process can be dependent on $I_s^{f,\delta}$ of P_f, and hence, there can be at most $N-1$ distinct direct dependency relations forming a transitive dependency relation. Let $P_{T_{rd}^{f,\delta}}(E_k)$ be the probability that any event in E_k may happen during $T_{rd}^{f,\delta}$. Then, $P_{rb}^{i,a}$ can be written as,

$$P_{rb}^{i,a}=P_{T_{rd}^{f,\delta}}(E_0 \cup E_1 \cup \ldots\ldots \cup E_{N-1}).$$

To calculate the probability $P_{rb}^{i,a}$, the probability that a direct dependency relation between two processes can be formed during $T_{rd}^{f,\delta}$, that is $P_{T_{rd}^{f,\delta}}(E_1)$, needs to be calculated first. The dependency relation between the processes are formed by message exchange. Since the interval for a process to send out two consecutive messages follows an exponential distribution with a rate λ_s, the expected number of messages sent out from a process during an interval $T_{rd}^{f,\delta}$ is $\lambda_s * T_{rd}^{f,\delta}$. When a process sends out one message, each of the other processes in the system can receive the message with the same probability of $\frac{1}{N-1}$. Hence, the probability for a process to receive at least one message sent out of $\lambda_s * T_{rd}^{f,\delta}$ messages is $1-(1-\frac{1}{N-1})^{\lambda_s * T_{rd}^{f,\delta}}$, which is the probability that a direct dependency relation can be formed from the process P_f to another process during $T_{rd}^{f,\delta}$. The probability that a transitive dependency relation with k direct dependency is formed during $T_{rd}^{f,\delta}$, that is $P_{T_{rd}^{f,\delta}}(E_k)$, can be calculated as the product of $P_{T_{rd}^{f,\delta}}(E_1)$.

Now, to complete the equation, the value of $T_{rd}^{f,\delta}$ needs to be calculated. Let Z denote the time interval of $I_s^{f,\delta}$ and X denote the random variable to represent the time interval between two successive failures of a process P_f. Since X has an exponential distribution with a rate $\lambda = \lambda_f * N$, the density function of X, $f_X(x)$ is, $f_X(x)=\lambda e^{-\lambda x}$, where $x \geq 0$, and the probability that a process fails during Z is, $F_X(Z)=P[X \leq Z]=1-e^{-\lambda Z}$. Hence, the expected time elapsed within Z before P_f fails under the condition that P_f fails during $I_s^{f,\delta}$ is calculated as,

$$T_{rd}^{Z}=\int_Z^0 t\frac{fx(x)}{Fx(Z)}\,dt=\frac{1}{\lambda}-\frac{Ze^{-\lambda Z}}{1-e^{-\lambda Z}} \text{ [1]}.$$

In reality, the process P_f may take some forced checkpoints during $I_s^{f,\delta}$ and hence, the value of $T_{rd}^{f,\delta}$ must be calculated for the actual checkpointing interval instead of the scheduled checkpointing interval; that is, the value of Z must be the average checkpointing interval value during $I_s^{f,\delta}$ instead of $I_s^{f,\delta}$ itself. Now, let $T^{*}{}_{rd}^{f,\delta}$ denote the actual rollback distance of P_f for the average checkpointing interval value during $I_s^{f,\delta}$. Then, the value of $T^{*}{}_{rd}^{f,\delta}$ is calculated as,

$$T^{*}{}_{rd}^{f,\delta}=\frac{1}{\lambda}-\frac{Te^{-\lambda T}}{1-e^{-\lambda T}},$$

where $T=\frac{T_s^{f,\delta}}{N_{fcp}^{f,\delta}+1}$. We now calculate the value of $T_{rd}^{i,a}$. According to our analysis model, when a process has to roll back, it always rolls back to its latest checkpoint. Let $T_s^{i,a}$ be the time interval of $I_s^{i,a}$. Then, the expected rollback distance during $I_s^{i,a}$ is the expected time elapsed within $T=\frac{T_s^{i,a}}{N_{fcp}^{i,a}+1}$ before P_i rolls back under the condition that P_i has to roll back during $I_s^{i,a}$. Let Y denote the random variable to represent the interval between two successive rollbacks of a process. Then, Y has an exponential distribution with a rate $\lambda_r = \lambda_f * N * P_{rb}^{i,a}$. Hence, in a similar way to calculate $T^{*\,f,\delta}_{rd}$, the value of $T^{i,a}_{rd}$ is calculated as,

$$T^{i,a}_{rd} = \frac{1}{\lambda_r} - \frac{T e^{-\lambda r T}}{1 - e^{-\lambda r T}}, \text{ where } T = \frac{T^{i,a}_{s}}{N^{i,a}_{frp} + 1}.$$

4. Performance Study

Extensive simulations were conducted to evaluate the effect of the proposed scheme on the system performance. For the evaluation, the proposed checkpoint scheduling scheme implemented on the checkpointing coordination scheme in [8] was simulated and the performance was compared with those of other checkpointing coordination schemes which do not employ the dynamic checkpoint scheduling such as the schemes suggested in [8] and [12].

4.1 Simulation Model

A distributed system consisting of 10 processes was simulated. During the normal computation, each of the processes sends out messages with an interval following an exponential distribution with a rate λ_s, and the receiver of each message is randomly selected. The communication delay between two processes follows an exponential distribution with a rate λ_d and 10.0 milliseconds are used as the value of $\frac{1}{\lambda_d}$ for the simulation. No particular network configuration or the implementation protocol is assumed for the message communication. For the schemes which do not employ the dynamic checkpoint scheduling, the time delay between two consecutive scheduled checkpointing of a process is assumed to follow a normal distribution with a mean N_c and a standard deviation of δ_c. With δ_c, the deviation of the local clock values of the processes running on the different nodes can be simulated, and δ_c is assumed to be $\frac{Nc}{2}$. For all of the simulated schemes, 100.0 milliseconds of constant checkpoint saving time is assumed.

A process may fail during its normal computation or checkpointing. If a process fails during the checkpointing, it must discard the current checkpoint and roll back to its previous checkpoint. The inter-arrival time of the failures of a process follows an exponential distribution with a rate of λ_f. For simplicity, we assume an instant recovery from a failure; that is, when a process fails, it instantly initiates a rollback coordination. To seek a consistent recovery line, a centralized rollback coordination scheme suggested in [14] is used. For the rollback coordination and restoring the selected checkpoint, 100.0 milliseconds of constant rollback time is assumed. As a performance index, the mean *computation loss* of a process, which is the average percentage of the process's execution time lost due to the checkpointing and the rollback-recovery compared to the total execution time, is used. The mean computation loss is calculated as follows:

$$ComputationLoss(\%) = \frac{\sum_{i=1}^{N} \frac{T^{i}_{cp} + T^{i}_{rs} + T^{i}_{rd}}{T^{i}_{E}}}{N} * 100,$$

where T^{i}_{cp} is the total amount of time spent to save checkpoints of a process *Pi*; T^{i}_{rs} is the total amount of time spent for the process *Pi* to coordinate the rollback propagation and restore the selected checkpoints; T^{i}_{rd} is the total amount of the computation lost due to the rollback of *Pi*; T^{i}_{E} is the total execution time of *Pi*; and N is the number of processes in the system. The mean computation loss of various schemes is thoroughly examined with the various values of the following system parameters: the mean scheduled checkpointing interval *Nc*, the mean failure inter-arrival time λ_f^{-1} and the mean message sending interval λ_s^{-1}.

4.2 Simulation Results

Figure 6 shows the performance of the schemes under the various values of the mean scheduled checkpointing interval, *Nc*, where λ_f^{-1} is 72000.0 seconds and λ_s^{-1} is 1.0 second. In the figure, Scheme P.A denotes the proposed scheme implemented using the analytic data and Scheme P.E denotes the proposed scheme using the empirical data. To reduce the computation burden for obtaining the analytic data, the value of $P_{T_W}^{i,a}$ was approximated to the value of $P_{T_W^a}(E_0 \cup E_1 \cup E_2)$. Scheme B represents the one suggested in [8]. Schemes W.2 and W.3 represent the ones suggested in [12], where Scheme W.2 allows the domino effect of two checkpointing intervals and Scheme W.3 allows the domino effect of three checkpointing intervals.

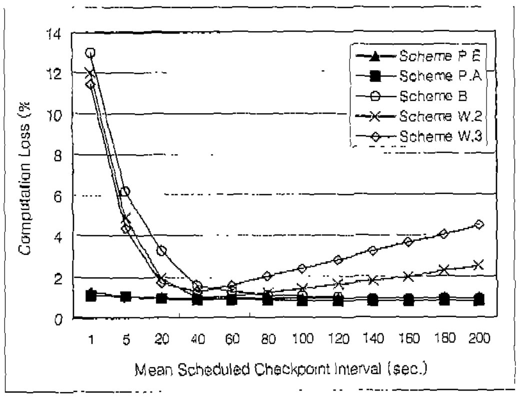

Fig. 6 Mean Computation Loss vs. Mean Scheduled Checkpointing Interval

For the proposed schemes, the given values of *Nc* are used only for the first scheduled checkpointing interval and the rest of the intervals are gradually adjusted based on the performance perceived by each process. Hence, Scheme P.A and Scheme P.E show the stable performance for the various values of the scheduled checkpointing interval. However, for the other schemes, the selection of the scheduled checkpointing interval value is very critical to their performance. When a short interval is used, the computation loss of Scheme B becomes sharply increased since Scheme B takes more forced checkpoints compared to Scheme W.2 and Scheme W.3. Meanwhile, when a long interval is used, Scheme W.3 shows the most sensitive response, since it produces the longer rollback distance for each rollback.

Figure7 shows the performance of the schemes under the various values of mean failure inter-arrival time, λ_f^{-1}, where the 120.0 seconds of *Nc* and the 1.0 second of λ^{-1} are used. The failure inter-arrival time mainly affects the number of rollbacks of the processes. For the smaller failure inter-arrival time, Scheme W.3 shows the worst performance compared to the other schemes, since the rollback distance for each rollback under Scheme W.3 is much longer than the others. For Scheme P.A and Scheme P.E, as the failure inter-arrival time becomes smaller, the processes tend to reduce their scheduled checkpointing interval so that the rollback distance for each rollback can be reduced. Because of this, the portion of the total amount of the rollback distance in the computation loss becomes stable, even though the total number of rollbacks becomes increased. However, the reduction in the scheduled checkpointing interval may slightly increase the total checkpointing time of the processes and it also increases the computation loss. The figure shows that such adjustment in the scheduled checkpointing interval in Schemes P.A and P.E are effective in reducing the total computation loss.

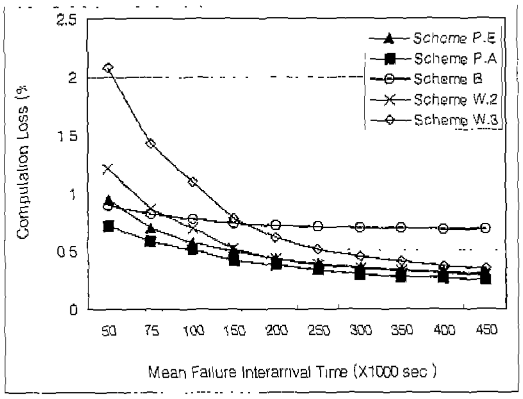

Fig. 7 Mean Computation Loss vs. Mean Failure Intervalarrival Time

Figure 8 shows the performance of the schemes under the various values of mean message sending interval, λ_s^{-1}, where *Nc* is 120.0 seconds and

λ_f^{-1} is 72000.0 seconds. The mean message sending interval affects the performance of the processes in two aspects: one is the probability of the rollback and the other is the number of forced checkpointing. With the short message sending interval, the computation loss of Schemes W.2 and W.3 increases because of the high probability of rollback involvement in case of a failure. The computation loss of Scheme B also increases with the short message sending interval since the number of forced checkpointing increases as more messages are exchanged between the processes. Under Schemes P.A and P.E, the processes adjust their scheduled checkpointing intervals considering both of the rollback probability and the number of forced checkpointing to reduce the total computation loss.

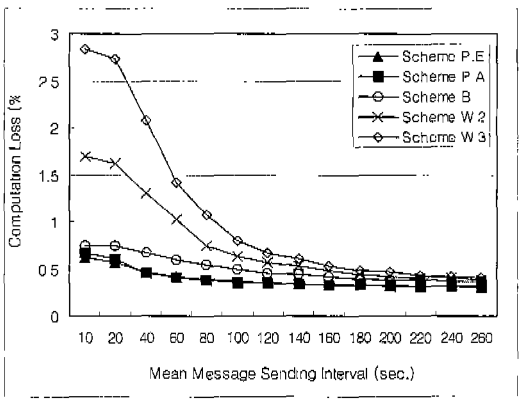

Fig. 8 Mean Computation Loss vs. Mean Message Sending Interval

5. Conclusions

In this paper, a new dynamic checkpointing scheduling scheme for a distributed system has been proposed. Each process in the proposed scheme selects the time interval for its scheduled checkpointing to reduce the overhead caused by both of the checkpointing and the rollback. Hence, the performance of the processes can be significantly improved over the other existing schemes which consider either the checkpointing overhead or the rollback overhead. Also, the checkpointing scheduling is performed dynamically, reflecting the current system states. As a result, the processes can show the very stable performance for the varying system parameter values.

References

[1] K.G. Shin, T. Lin, and Y. Lee, "Optimal checkpointing of real-time tasks," *IEEE Trans. on Computers*, Vol. C-36, No. 11, pp. 1328-1341, 1987.

[2] A.N. Tantawi and M. Ruschitzka, "Performance analysis of checkpointing strategies," *ACM Trans. on Computer Systems*, Vol. 2, No. 2, pp. 123-144, 1984.

[3] A. Ziv and J. Bruck, "Analysis of checkpointing schemes for multiprocessor systems," in *Proc. of the 13th Symp. on Reliable Distributed Systems*, pp. 52-61, 1994.

[4] A.P. Sistla and J.L. Welch, "Efficient distributed recovery using message logging," in *Proc. of the 8th ACM Symp. on Principles of Distributed Computing*, pp. 223-238, 1989.

[5] J.L. Kim and T. Park, "An efficient algorithm for checkpointing recovery in distributed systems," *IEEE Trans. on Parallel and Distributed Systems*, Vol. 4, No. 8, pp. 955-960, 1993.

[6] B.L. Randell, P.A. Lee, and P.C. Treleaven, "Reliability issue in computing system design," *ACM Computing Surveys*, Vol. 2, pp. 123-166, 1978.

[7] T. Park and J.L. Kim, "Domino-effect free checkpointing recovery in distributed systems," in *Proc. of the 7th Int'l Conf. on Parallel and Distributed Computing Systems*, pp. 497-502, 1994.

[8] D. Briatico, A. Ciuffoletti, and L. Simoncini, "A distributed domino-effect free recovery algorithm," in *Proc. of the 4th Symp. on Reliability in Distributed Software and Database Systems*, pp. 207-215, 1984.

[9] R. Koo and S. Toueg, "Checkpointing and rollback-recovery for distributed systems," *IEEE Trans. on Software Engineering*, Vol. SE-13, No. 1, pp. 23-31, 1987.

[10] Y. Tamir and C.H. Sequin, "Error recovery in multicomputers using global checkpoints," in *Proc. of the 14th IEEE Symp. on Fault-Tolerant Computing*, pp. 32-41, 1984.

[11] K. Venkatesh, T. Radhakrishan, and H.F. Li, "Optimal checkpointing and local recording for domino-free rollback recovery," *Information Processing Letters*, Vol. 25, pp. 295-303, 1987.

[12] Y.M. Wang and W.K. Fuchs, "Lazy checkpoint coordination for bounding rollback propagation," in *Proc. of the 12th Symp. on Reliable Distributed Systems*, pp. 78-85, 1993.

[13] R.D. Schelichting and F.B. Schneide, "Fail-stop processors: An approach to designing fault-tolerant computing systems," *ACM Trans. of Computer Systems*, Vol. 1, No. 3, pp. 222-238, 1983.

[14] B. Bhargava and S. Lian, "Independent checkpointing and concurrent rollback for recovery in distributed systems - An optimistic approach," in *Proc. of the 7th IEEE Symp. on Reliable Distributed Systems*, pp. 3-12, 1988.

박 태 순

1983년 ~ 1987년 서울대학교 전자계산기공학과 학사. 1988년 ~ 1989년 Texas A&M University 석사. 1990년 ~ 1994년 Texas A&M University 박사. 1995년 ~ 1997년 세종대학교 정보처리학과 전임강사. 1998년 ~ 현재 세종대학교 컴퓨터공학과 조교수. 관심분야는 분산 시스템, 결함내성 시스템, 분산 데이타베이스 시스템.