Analysis of failure recovery schemes for distributed shared-memory systems

J. H. Kim and N. H. Vaidya

Abstract: Checkpoint and rollback recovery is a technique used to minimise the loss of computation when failures occur. When a process rolls back and re-executes from the last checkpoint, the *cost* (loss) incurred by redoing the lost computation may be larger than that to execute the original computation. In addition to completion time delay, other performance metrics (e.g. user's satisfaction in real-time on online transaction applications) may also degrade by unexpected failure and recovery. The paper determines how *redo* overhead factor for unexpected execution overhead affects the performance of recovery scheme. It analyses the performance of three recoverable schemes (incorporating *redo* overhead factor): multiple fault-tolerant scheme using checkpointing and rollback recovery, single fault-tolerant scheme, and a two-level scheme.

1 Introduction

Checkpoint and rollback recovery is a technique used to minimise the loss of computation when failures occur. A checkpoint is a state of application stored on a stable storage. The application periodically saves checkpoints, and can recover from a failure by rolling back to the checkpoint [1]. When a process rolls back and re-executes from the last checkpoint, the *cost* (loss) incurred by redoing the lost computation may be larger than that to execute the original computation. In addition to completion time delay, other performance metrics (e.g. user's satisfaction in real-time or online transaction applications) may also degrade by unexpected failure and recovery.

For an example, consider an airline reservation system. If the system fails temporarily, transactions executed during t time units will re-execute. An airline company will lose a lot of money by suspending reservation during the t time units. The monetary loss by stopping reservation for t time units may be much more serious than the computational loss of t time units. As another example, we can consider real-time systems, e.g. embedded systems and a critical information systems that must be highly available. A timing constraint or deadline is important in these real-time systems. A task completed after a deadline has little value. Redoing computation lost due to failure may cause a task to miss its deadline. In many cases, an user's perception of a system may become much worse as the delay in completing a task increases. Therefore the cost of redoing computation lost due to failure may be much larger than the lost CPU time.

IEE Proceedings online no. 19990420
DOI: 10.1049/ip-cdt:19990420

Paper first received 25th June 1998, and in revised form 3rd March 1999

J.H. Kim is with the School of Information and Computer Engineering, Ajou University, Suwon 442-749, South Korea
E-mail: jaikim@madang.ajou.ac.kr

N.H. Vaidya is with the Department of Computer Science, Texas A&M University, College Station TX 77843-3112, U.S.A.
E-mail: vaidya@cs.tamu.edu

We analyse three recoverable schemes incorporating *redo* overhead factor. The three schemes analysed in this paper are summarised as follows:

- *Multiple fault-tolerant scheme:* Checkpoint and rollback recovery scheme can be used to reduce the loss of computation on failure. When the application executes on multiple processors a consistent checkpointing scheme is used to save a global consistent state of the multiprocess application [2–4].

  An arbitrary number of failures can be tolerated by rolling the application back to the most recent consistent checkpoint.
- *Single fault-tolerant scheme:* Our recoverable distributed shared memory (DSM) scheme [5, 6] is an example of a single fault-tolerant scheme. This scheme maintains at least two copies for each page in a DSM to recover from single-node failure. Additional cost is incurred to maintain at least two copies for each page. Multiple failures require restart of the application.
- *Two-level recovery scheme:* We consider a two-level recovery scheme obtained by combining *multiple fault-tolerant checkpointing scheme* and *single fault-tolerant scheme* presented in [5, 6]. With this two level scheme, a single-node failure (more probable) can be recovered with the single fault-tolerant scheme, while multiple-node failure (less probable) can be recovered using a global consistent checkpoint.

2 Related work

For a transaction-oriented system, Chandy *et al.* [1] measure the time to *redo* the transactions arrived during t time units by using *compression factor*. They assume underloaded transaction system and do not consider the cost of delaying the newly arrived transactions after failure.

Many researchers present cost analysis for the rollback recovery scheme [1, 7, 8], rollforward scheme [9, 10], and replicating data [11]. Some papers analyse cost for recovery schemes using volatile memory [7, 11, 12], while many other papers present recovery scheme using volatile memory without any analysis [5, 6, 13–17]. Chandy *et*

al. [1], Young [8], and Vaidya [18] present methods to compute optimal checkpoint interval to minimise expected (average) execution time. In this paper, we compute optimal checkpoint interval including the 're-do' overhead factor and the two-level recovery scheme.

Theel and Fleisch analyse the costs for read and write operations, and availability for the boundary-restricted protocol [11]. They also present dynamic boundary-restricted protocol that can change the range (boundaries) of the number of cached copies to reduce operation cost while maintaining desired data availability.

Vaidya [7] presents a two-level recovery scheme that tolerates the more probable failures with low performance overhead, while the less probable failures may possibly incur a higher overhead. By minimising overhead for the more frequently occurring failure scenarios, the two-level approach can achieve lower performance overhead (on average) as compared to single-level recovery schemes. We add the *redo* overhead factor, and consider optimal checkpoint interval for the second level.

3 Performance analysis

This Section presents performance analysis for the three recovery schemes described in the previous Section. For this analysis we incorporate a *redo* overhead factor k that is defined as the relative cost of additional computation needed due to failure.

3.1 Multiple fault-tolerant scheme

Expected (average) execution time (denoted as Γ) of a single checkpoint interval with *redo* overhead factor $k=1$ is evaluated in [8, 18]. We consider expected cost Γ with *redo* overhead factor $k>1$, and analyse optimal checkpoint interval by varying *redo* overhead factor k. Using analysis similar to [18], the expected cost Γ is computed as follows:

$$\Gamma = (1-k)(T+C) + k\lambda^{-1}e^{\lambda R}(e^{\lambda(T+C)} - 1) \qquad (1)$$

where T is the amount of *useful* computation per checkpoint interval, C is checkpoint overhead, R is overhead of rollback to a checkpoint, and λ is the aggregate failure rate of all nodes in the system. (*Useful* computation excludes the cost spent on checkpointing and rollback recovery.) Failures are assumed to be governed by a Poisson process. Note that *redo* overhead factor k is only included for the computation overhead due to failures. Refer to Fig. 1 which shows the time required for different operations and also their costs.

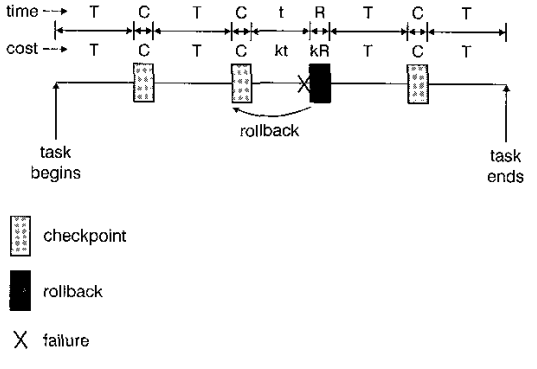

Fig. 1 *Checkpoint and rollback recovery scheme*

Let $G(t)$ denote the expected cost required to perform t units of *useful* computation. Then we define *overhead ratio* r as [18]

$$r = \lim_{t\to\infty}\frac{G(t)-t}{t} - \lim_{t\to\infty}\frac{G(t)}{t} - 1 \qquad (2)$$

In this analysis, overhead ratio r is given by

$$r = \frac{\Gamma}{T} - 1 \qquad (3)$$

To compute the optimal value of $T(T_{opt})$ that minimises the overhead ratio r

$$\frac{\partial r}{\partial T} = 0 \qquad (4)$$

Using the expansion of $e^{\lambda(T+C)}$ and $e^{\lambda R}$ as far as the second-degree term

$$e^{\lambda(T+C)} \simeq 1 + \lambda(T+C) + \frac{\lambda^2(T+C)^2}{2} \qquad (5)$$

$$e^{\lambda R} \simeq 1 + \lambda R + \frac{\lambda^2 R^2}{2} \qquad (6)$$

By approximation, ignoring the λ^3 term, and simplification, the optimal T is obtained as

$$\Rightarrow T \simeq \sqrt{\frac{2C}{\lambda k}\left(1 + \lambda k R + \frac{\lambda k C}{2}\right)} \qquad (7)$$

$$\Rightarrow T \simeq \sqrt{\frac{2C}{\lambda k}} \qquad (8)$$

when $\lambda kR \ll 1$ and $\lambda kC \ll 1$. We now present numerical examples.

Example 1 Assume the following parameters: checkpoint overhead C is two time units, recovery overhead R is two time units, failure rate λ is 0.01 per time unit. Fig. 2 shows overhead ratio r by varying checkpoint period T. Overhead ratio is minimum at $T=18.7$, 13.6, 10.0 for $k=1$, 2, 4, respectively. The checkpoint period T that minimises the overhead ratio r is close to computed approximation value T_{opt}: $T_{opt}=20.0$, 14.1, 10.0 for $k=1$, 2, 4, respectively.

Example 2: Assume that failure rate λ is 0.001 per time unit, and the other parameters are the same as the previous example. Fig. 3 shows overhead ratio r by varying checkpoint period T. Overhead ratio is minimum at $T=61.9$, 44.1, 31.4 for $k=1$, 2, 4, respectively. The checkpoint period T that minimises the overhead ratio r is also close to

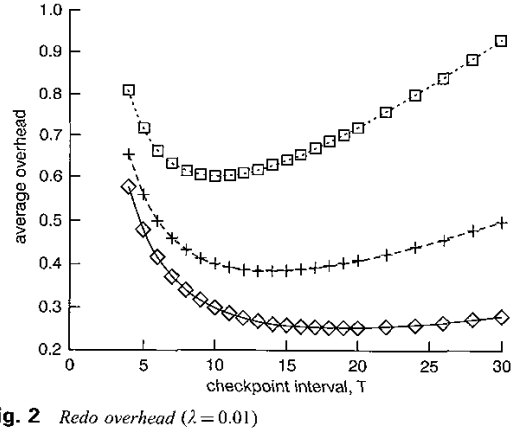

Fig. 2 *Redo overhead* ($\lambda=0.01$)
$-\diamond-$ $k=1$; $-\square-$ $k=4$ $-+-$ $k=2$

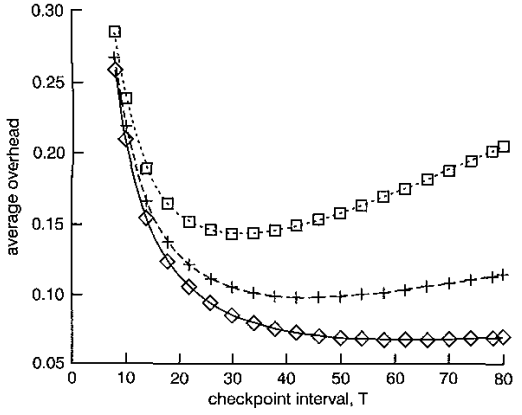

Fig. 3 *Redo overhead* ($\lambda = 0.001$)
$-\diamond- k=1$; $-\square- k=4$ $-+- k=2$

computed approximation value T_{opt}:$T_{opt} = 63.2$, 44.7, 31.6 for $k = 1$, 2, 4, respectively.

We observe that the optimal checkpoint interval T_{opt} decreases as *redo* overhead factor k increases. This is intuitive, because larger k implies that the cost incurred by a failure is high if the checkpoint interval is large.

3.2 Single fault-tolerant scheme

Our single fault-tolerant scheme [5, 6] maintains at least two copies for each page to recover from single-node failure. Additional cost is incurred to maintain at least two copies for each page when a node executes *release* operation. To simplify analysis, we assume that the failure-free overhead of the single fault-tolerant scheme to maintain at least two copies for each page is a multiplicative factor α. If the failure-free execution time of a task without using a recovery scheme is γ, the failure-free execution time of the task using the single fault-tolerant scheme will be $\alpha\gamma$. We also assume that recovery overhead to recover from a single fault is a constant R. We assume that if another failure occurs before system recovers from a failure, a restart is required. (This assumption is somewhat pessimistic.) A single fault-tolerant scheme is illustrated in Fig. 4; the Figure shows the time and cost required for different operations.

Let $f(\alpha\gamma)$ denote the expected *time* (not cost) required to perform γ units of *useful* computation (we determine the expected cost later). Then $f(t+\varepsilon) - f(t)$ is the time required to perform ε time units of computation starting from time t. Let $E(x)$ denote the expected amount of time

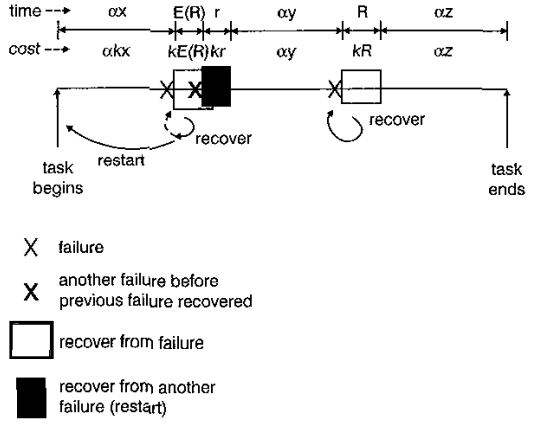

Fig. 4 *Single fault-tolerant scheme*

during an interval of length x before a failure occurs, given that a failure occurs sometime during the interval. Then

$$E(x) = \int_0^x t \frac{\lambda e^{-\lambda t}}{1 - e^{-\lambda x}} dt = \lambda^{-1} - \frac{x e^{-\lambda x}}{1 - e^{-\lambda x}} \tag{9}$$

Now, consider a small interval of length ε, starting at time t. When ε is small, the probability of failure in that interval is $\lambda\varepsilon$. There are three cases, as follows.

(i) No fault occurs during ε (probability is $1 - \lambda\varepsilon$): ε time units are required to complete the interval.
(ii) A fault occurs during ε, but the fault is recovered without another fault during the recovery time R (probability is $\lambda\varepsilon e^{-\lambda R}$): $\varepsilon + R$ time units are required to complete the interval. (ε is sufficiently small value so that we ignore the probability of another fault occuring in the ε interval.)
(iii) A fault occurs during ε, and another fault occurs before recovering from the first failure (probability is $\lambda\varepsilon(1 - e^{-\lambda R})$): In this case, $E(\varepsilon) + E(R) + r + f(t) + (f(t+\varepsilon) - f(t))$ time units are required to complete the interval; $E(\varepsilon) + f(t)$ is required for *redoing* the lost computation, $E(R)$ for recovering from the first failure, r for a restart and $f(t+\varepsilon) - f(t)$ for the original ε unit computation. We assume that $r = 0$.

Thus, $f(t+\varepsilon) - f(t)$ is obtained as

$$\begin{aligned} f(t+\varepsilon) - f(t) = {} & (1 - \lambda\varepsilon)\varepsilon \\ & + (\lambda\varepsilon e^{-\lambda R})(R + \varepsilon) \\ & + (\lambda\varepsilon(1 - e^{-\lambda R})) \\ & \times [E(\varepsilon) + E(R) + f(t) + f(t+\varepsilon) - f(t)] \end{aligned} \tag{10}$$

On simplification and taking a limit as ε approaches 0, we obtain

$$\begin{aligned} \lim_{\varepsilon \to 0} \left[\frac{f(t+\varepsilon) - f(t)}{\varepsilon} \right] &= 1 + \lambda e^{-\lambda R} R + \lambda(1 - e^{-\lambda R}) E(R) \\ &\quad + \lambda(1 - e^{-\lambda R}) f(t) \\ \Rightarrow \frac{\partial f(t)}{\partial t} &= A + B f(t) \end{aligned} \tag{11}$$

where $A = 1 + \lambda e^{-\lambda R} R + \lambda(1 - e^{-\lambda R})\ E(R)$ and $B = \lambda(1 - e^{-\lambda R})$. By calculus

$$f(t) = \frac{A}{B} e^{Bt} - \frac{A}{B} \tag{12}$$

Now we determine the average cost of executing t units of useful computation. Observe that, out of $f(\alpha\gamma)$, average time spent on unexpected execution (due to failure) is $f(\alpha\gamma) - \alpha\gamma$. Thus, the average cost due to *redo* is $k(f(\alpha\gamma) - \alpha\gamma)$. Therefore the average cost required to perform γ units of *useful* computation, denoted as $g(\alpha\gamma)$, is

$$g(\alpha\gamma) = \alpha\gamma + k(f(\alpha\gamma) - \alpha\gamma) \tag{13}$$

$$\Rightarrow g(\alpha\gamma) = \alpha\gamma(1 - k) + k f(\alpha\gamma) \tag{14}$$

$$\Rightarrow g(\alpha\gamma) = \alpha\gamma(1 - k) + k\left(\frac{A}{B} e^{B\alpha\gamma} - \frac{A}{B}\right) \tag{15}$$

The *average overhead* is evaluated as a fraction of γ (task length)

$$r = \frac{g(\alpha\gamma)}{\gamma} - 1 \tag{16}$$

$$\Rightarrow r = \alpha(1 - k) + \frac{k}{\gamma}\left(\frac{A}{B} e^{B\alpha\gamma} - \frac{A}{B}\right) - 1 \tag{17}$$

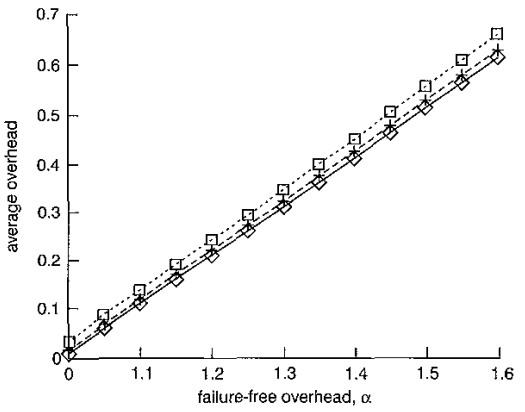

Fig. 5 *Overhead for single fault-tolerant scheme*
–◇– $k=1$; –□– $k=4$ –+– $k=2$

Fig. 5 shows overhead ratio r by varying the failure-free overhead factor (α) for the single fault-tolerant scheme. We use a short task length ($\gamma=80$) for this analysis. We assume that task length γ is 80, recovery overhead R is 0.6 time units, failure rate λ is 0.01 per time unit. The overhead of single fault-tolerant scheme is lower than that of the checkpoint scheme (shown in Fig. 2) using optimal checkpoint interval, when the failure-free overhead factor α for the single fault-tolerant scheme is less than 1.25 (at $k=1$), 1.36 (at $k=2$), and 1.55 (at $k=4$). Our previous research [6] shows that the failure-free overhead α for a single fault-tolerant scheme is less than 1.25 in many applications. Another observation is that the single fault-tolerant DSM scheme is not too sensitive to *redo* overhead factor, if the task length is short and the failure rate is not high, while the checkpoint scheme is more sensitive to *redo* overhead factor. Thus, in particular, our single fault-tolerant scheme is better for high *redo* overhead factor, for short tasks.

However, our single fault-tolerant DSM scheme is not good when the task length is very long. As the task length increases, the probability of rolling back to the start point of the task owing to multiple-failure becomes large, while the average overhead of checkpoint scheme is essentially independent of the task length. Fig. 6 shows the average overhead by varying task length. We use fixed failure-free overhead ($\alpha=1.1$). As task length increases, the average overhead increases. *Redo* overhead factor affects the average overhead more, as the task length increases.

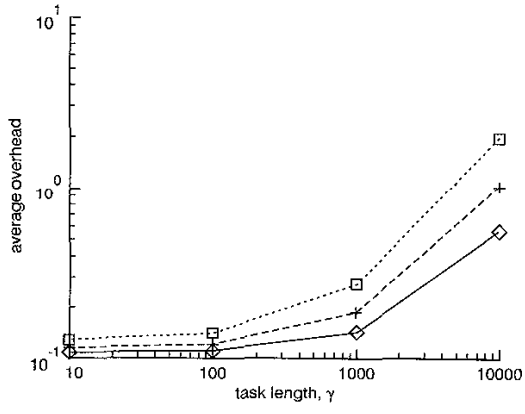

Fig. 6 *Overhead for single fault-tolerant scheme*
–◇– $k=1$; –□– $k=4$ –+– $k=2$

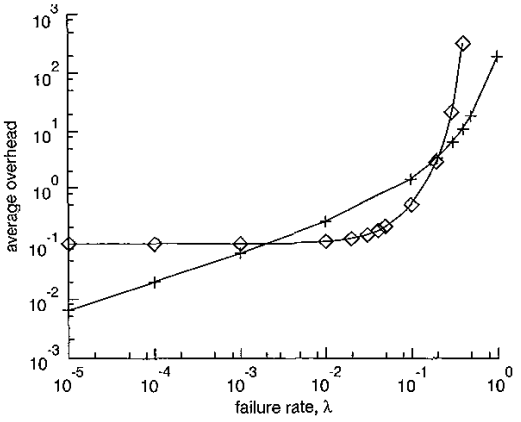

Fig. 7 *Single fault-tolerant against checkpoint scheme*
—◇—Single fault tolerant
—+—checkpoint

Fig. 7 shows the average overhead for a single fault-tolerant DSM scheme and checkpoint scheme by varying failure rate λ. We use $k=1$, $\alpha=1.1$, and $\gamma=80$ for this analysis. For the single fault-tolerant scheme at low failure rate, the average overhead is approximately equal to $\alpha-1$ because the single fault-tolerant scheme is enough to recover most faults. However, the average overhead increases rapidly when the failure rate is high. Overhead of the checkpoint scheme is lower than the single fault-tolerant scheme for small λ. As the failure rate becomes moderately large, the single fault-tolerant scheme performs better. The overhead of the single fault-tolerant scheme is still near $\alpha-1$ because most failures can be recovered by the single fault-tolerant scheme, while the overhead of the checkpoint scheme increases more rapidly. At a higher failure rate, again the checkpoint scheme performs better, because the single fault-tolerant scheme suffers from frequent restarting due to multiple near-simultaneous failures.

In general, the single fault-tolerant scheme performs better for low failure-free overhead, short task length, and/or low failure rate. However, if the task length is long and/or failure rate is very high, it is highly possible that another failure will occur before recovering from the previous failure. When more than one failure occurs, the task has to restart from the initial point because single fault-tolerant scheme can recover a single failure only. To solve this problem, we consider a two-level scheme in the following Section.

3.3 Two-level scheme

The two-level scheme periodically takes checkpoints to allow recovery from arbitrary number of failures. Between checkpoints, the single fault-tolerant scheme is used to allow quick recovery from single failures. Fig. 8 illustrates

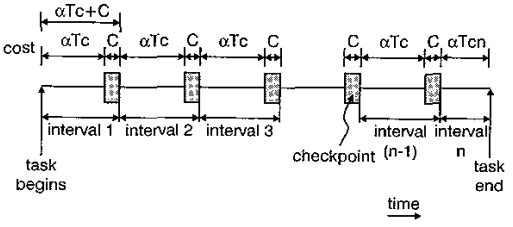

Fig. 8 *Failure-free execution*

a failure-free execution of two-level scheme. To evaluate expected execution time, we assume that the amount of useful computation in a checkpoint interval is T_c, and checkpoint overhead is C. Each checkpoint interval ends with a checkpoint except the last interval. There are $\lceil \frac{\gamma}{T_c} - 1 \rceil$ intervals of length $\alpha T_c + C$ that requires the cost of $g(\alpha T_c + C)$ per interval, and the last interval of length $\alpha\gamma - \lceil \left(\frac{\gamma}{T_c}\right) - 1 \rceil \alpha T_c$ (αT_{cn} in Fig. 8) that requires the cost of $g(\alpha\gamma - \left\lceil \frac{\gamma}{T_c} - 1 \right\rceil \alpha T_c)$. Thus, the expected task completion cost Γ is

$$g(\alpha T_c + C)\left\lceil \frac{\gamma}{T_c} - 1 \right\rceil + g\left(\alpha\gamma - \left\lceil \frac{\gamma}{T_c} - 1 \right\rceil \alpha T_c\right) \tag{18}$$

$g(t)$ is defined in section 3.2.

As an example, Fig. 9 shows the average overhead by using the following parameters: failure-free overhead for the single fault-tolerant scheme is 1.1, *redo* overhead factor is 1.0, overhead of checkpoint C is 2.0, rollback overhead by single fault-tolerant scheme R is 0.6, rollback overhead to checkpoint R_c is 2.0, failure rate is 0.1, length of task is 80. From Fig. 9, observe that the average overhead is minimised when the amount of useful computation in a checkpoint interval T_c is 20.

To compare the performance of the two-level scheme with the one-level checkpoint scheme, we use a very long

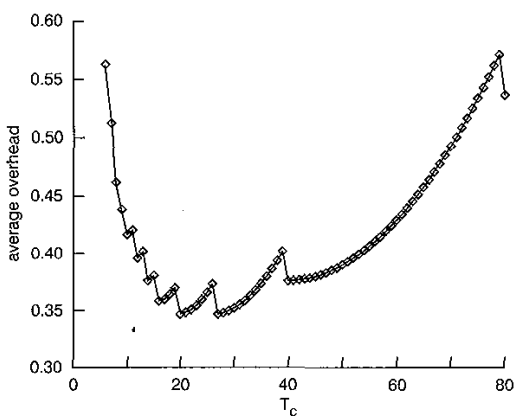

Fig. 9 *Minimum achieved when $T_c = 20$*
—◇—two level: $k = 1$

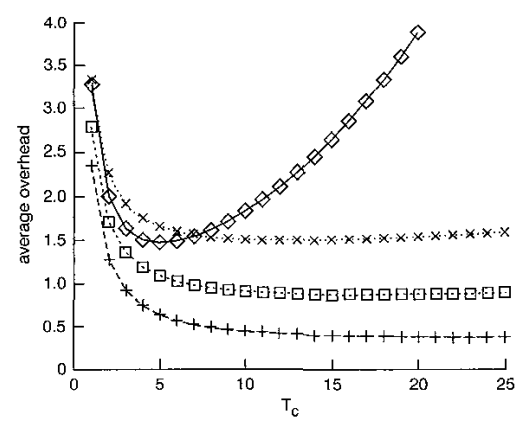

Fig. 10 *Minimum achieved when $T_c = 24.9/\alpha$*
—◇—checkpoint
—+—two level: $\alpha = 1.1$
—□—two level: $\alpha = 1.5$
—×—two level: $\alpha = 2.0$

length of task ($\gamma = 1{,}000{,}000$) (other parameter are the same as the previous example). As shown in Fig. 10 (checkpoint denotes checkpointing scheme, and two-level: alpha $= \alpha$ denotes a two-level scheme with the failure-free overhead of α for the single fault-tolerant scheme), the minimum overhead of the two-level scheme with $\alpha = 2.0$ is comparable with that of one-level checkpoint scheme. In this example, the two-level scheme with $\alpha < 2.0$ is better than the one-level checkpoint scheme. The $\alpha_{critical}$ where two schemes require same overhead will decrease as λ decreases because the overhead of the checkpoint scheme decreases.

The two-level recovery scheme includes the single fault-tolerant (one-level) scheme as a special case. When $T_c = \gamma$, the two-level scheme is identical to the single fault-tolerant scheme. The two-level scheme can be used when γ is long, λ is high, and/or R is long, because in these cases it is highly possible that processes using the single fault-tolerant scheme may restart owing to multiple-failure (another failure occurring before recovering from a previous failure).

3.4 Optimal checkpoint interval

Now we compute the optimal checkpoint interval of two-level scheme approximately. In our case, the *first-level* recovery scheme is the single fault-tolerant scheme, and the *second-level* scheme is the checkpointing scheme. The failure from the point of view of the checkpointing scheme is the failure that cannot be recovered by the *first-level* recovery scheme. If another failure occurs before a failure is recovered, this is considered as 'failure' in the *second level*. Thus failure rate λ_2 of the *second level* is computed as follows:

$$\lambda_2 = \lim_{e \to 0} \frac{(1 - e^{-\lambda e})(1 - e^{-\lambda R})}{\varepsilon} \tag{19}$$

$$\Rightarrow \lambda_2 = \lim_{e \to 0} \frac{\lambda_\varepsilon (1 - e^{-\lambda R})}{\varepsilon} \tag{20}$$

$$\Rightarrow \lambda_2 = \lambda(1 - e^{-\lambda R}) \tag{21}$$

This is only an approximation because the second-level failures are not exponentially distributed. If the second-level failures were exponentially distributed, the probability of failure in the interval [0, t] would be $1 - e^{-\lambda}(1 - e^{-\lambda R})t$. However, actual probability of failure in the interval [0, t] is $(1 - e^{-\lambda t})$ $(1 - e^{-\lambda R})$. An approximation of the optimal checkpoint interval for the two-level scheme (obtained using equation for T_{opt}, and assuming that second level failures as exponentially distributed with mean interval $1/\lambda_2$) is

$$T_{opt2} = \sqrt{\frac{2C}{\lambda_2 k}} = \sqrt{\frac{2C}{\lambda(1 - e^{-\lambda R})k}} \tag{22}$$

As an example, in Fig. 10 the overhead of two-level scheme is minimum at $T_c = 24.9/\alpha$. The computed optimal checkpoint interval T_{opt2} for two-level scheme is

$$T_{opt2} = \sqrt{\frac{2C}{\lambda_2 k}} = \sqrt{\frac{2C}{\lambda(1 - e^{-\lambda R})k}} = \sqrt{\frac{2 \times 2}{0.1 \times (1 - e^{-0.1 \times 0.6})}} = 26.2 \tag{23}$$

$T_{opt2} = 26.2$ is close to the actual optimal checkpoint interval (24.9).

4 Conclusion

This paper evaluated how *redo* overhead factor affected the cost of recoverable DSM. We also analysed optimal checkpoint interval by varying the *redo* overhead factor. We analysed and compared the performance of three recoverable schemes (multiple fault-tolerant scheme, single fault-tolerant scheme, and two-level scheme) incorporating the *redo* factor.

In general the single fault-tolerant scheme presented in [5, 6] is better for a low failure-free overhead, short task length, and/or moderate failure rate. However, if task length is long and/or failure rate is high, it is highly possible that another failure will occur before recovering from a previous failure. When more than one failure occurs, the task has to restart from the initial point because the single fault-tolerant scheme can recover a single failure only. To solve this problem, we used the two-level recovery scheme. The two-level scheme tends to perform better than the checkpointing scheme unless the failure-free overhead is large and/or the failure rate is very small.

7 Acknowledgments

This work was supported in part by the National Science Foundation under grant MIP-9502563, and Ajou University research fund 1998. We thank the referees for their insight.

8 References

1 CHANDY, K., BROWNE, J., DISSLY, C., and UHRIG, W.: 'Analytic models for rollback and recovery strategies in data base systems,' *IEEE Trans.*, 1975, **SE-1**, pp. 100–110
2 CHANDY, K., and LAMPORT, L.: 'Distributed snapshots: Determining global states in distributed systems,' *IEEE Trans.*, 1985, **3**, pp. 63–75
3 ELNOZAHY, E., JOHNSON, D., and ZWAENEPOEL, W.: 'The performance of consistent checkpointing', Proceedings of the eleventh symposium on *Reliable Distributed Systems*, Houston, Texas, USA, Oct. 1992, pp. 39–47
4 KOO, R., and TOUCG, S.: 'Checkpointing and rollback-recovery for distributed systems,' *IEEE Trans.*, 1987, **SE-13**, pp. 23–31
5 KIM, J.-H., and VAIDYA, N. H.: 'Recoverable distributed shared memory using the competitive update protocol', Proceedings of the Pacific Rim international symposium on *Fault-Tolerant Systems*, Newport Beach, California, USA, Dec. 1995, pp. 152–157
6 KIM, J.-H. and VAIDYA, N.H.: 'Single fault-tolerant distributed shared memory using competitive update,' *Microprocess. Microsyst.*, 1997, **21**, pp. 183–196
7 VAIDYA, N.H.: 'A case for two-level recovery schemes,' *IEEE Trans.*, 1998, **C-47**, pp. 656–666
8 YOUNG, J.: 'A first order approximation to the optimum checkpoint interval,' *Commun. ACM*, 1974, **17**, pp. 530–531
9 LONG, J., FUCHS, W., and ABRAHAM, J.: 'Forward recovery using checkpointing in parallel systems', Proceedings of international conference on *Parallel Processing*, St. Charles, Illinois, USA, Aug. 1990, pp. 272–275
10 PRADHAN, D.K., and VAIDYA, N. H.:'Roll-forward checkpointing scheme: A novel fault-tolerant architecture,' *IEEE Trans.*, 1994, **C-43**, pp. 1163–1174
11 THEEL, O., and FLEISCH, B.: 'Design and analysis of highly available and scalable coherence protocols for distributed shared memory systems using stochastic modeling', Proceedings of the international conference on *Parallel Processing*, Oconomowoc, Wisonsin, USA, Aug. 1995, **1**, pp. 126–130
12 THEEL, O., and FLEISCH, B.: 'Analysis of a fault-tolerant coherence protocol for distributed memory systems under heavy write load'. Proceedings of the 1995 Pacific Rim international symposium on *Fault-Tolerant Systems*, Newport Beach, California, USA, Dec. 1995, pp. 146–151
13 BANATRE, N., GEFFLAUT, A., and MORIN, C.: 'Tolerating node failures in cache only memory architectures'. Technical report 853, INRIA, 1994
14 BROWN, L., and WU, J.: 'Dynamic snooping in a fault-tolerant distributed shared memory', Proceedings of the 14th international conference on *Distributed computing systems*, Poznan, Poland, June 1994, pp. 218–226
15 FUCHI, T., and TOKORO,M.: 'A mechanism for recoverable shared virtual memory'. University of Tokyo (manuscript) 1994
16 KERMARREC, A.-M., CABILLIC, G., GEFFLAUT, A., MORIN, C., and PUAUT, I.: 'A recoverable distributed shared memory integrating coherence and recoverability' Proceedings of the 25th international symposium on *Fault-tolerant Computing*, Pasadena, California, USA, June 1995, pp. 289–298
17 STUMM, M., and ZHOU, S.: 'Fault tolerant distributed shared memory algorithms', Proceedings of the international conference on *Parallel and Distributed Processing*, Dallas, Texas, USA, Dec. 1990, pp. 719–724
18 VAIDYA, N.H.: 'Impact of checkpointing latency on overhead ratio of a checkpointing scheme,' *IEEE Trans.*, 1997, **C-46**, pp. 942–947