A Damage Assessment Model in a Distributed System

Parimal Kumar Giri
Department of CSE
APEX Institute of Technology & Management
Bhubaneswar, India

Satya Ranjan Mohapatra
Department of CSE
Gandhi Institute For Technology
Bhubaneswar, India

ABSTRACT
Error propagation is unavoidable due to imperfect detection mechanisms and random inter-process communications; it could give rise to contaminated checkpoints, which, in turn, result in unsuccessful rollbacks. To encounter the problem of error propagation, a damage assessment model is discussed to optimize the correctness of saved checkpoints under various circumstances. The algorithm is based on an equivalence classes between pairs of successive checkpoints of a process which allows us, in some cases, to advance the recovery line of the computation without forcing checkpoints in other processes. This is well-suited for autonomous and heterogeneous environments, where each process does not know any private information about other processes and private information of the same type of distinct processes is not related.

General Terms
Index based checkpointing algorithm, Damage assessment model,, learning technique, Rollback Recovery.

Keywords
Contaminated Check-points; Error Propagation; Nonlinear Integer Programming; Damage Assessment Model; Global Checkpoint; Rollback Recovery; Index-based Check-pointing.

1. INTRODUCTION
Index-based Checkpointing algorithm for distributed systems with the aim of reducing the total number of checkpoints while ensuring that each checkpoint belongs to at least one consistent global checkpoint (or recovery line) [1].Consider a distributed system consisting of n processes, one process per processor, denoted by $\{P_0, P_1, P_2, \cdots, P_{n-1}\}$ and a set of bidirectional channels on an arbitrary network topology. There is no common clock, shared memory or central coordinator. Message passing is the only mode of communication between any pair of processes. Any process can initiate Check-pointing. Index-based algorithms associate each local checkpoint with a sequence number and try to enforce consistency among local checkpoints with the same sequence number. Check-pointing and rollback-recovery are well-known techniques for providing fault-tolerance in distributed systems [1,2].

1.1 Damage Assessment
The method of damage assessment is developed by probabilistically characterizing the interval between the occurrence and the detection of an error using parameters associated with faults and errors. A fault is defined as any defect capable of causing potential damage, or any deviation from the normal state of a computing system. An error is deviation from the specification of the program running on a computing system.

Consider a multi-module computing system, where the processes communicate with one another via message passing. Each process is assume to run on a separate module and can be represented by a Digraph, D=(V,E) where V= { $v_1,v_2,.....v_n$ } and E= { e_{ij} : 1<=(i, j)<=n} denotes the set of nodes(i,e modules) and set of directed edges (communication channels).A module is said to be faulty if it contains faults, and contaminated if contains errors. Let T_i^F , the v_i's faulty time and T_i^C ,the v_i's contamination time ,then damage assessment can be viewed as the estimation of the distributions of all modules contamination times [3].

1.2 Fault Detection Mechanism
Faults are detected directly by periodic diagnostics or fault detection mechanisms, such as self-checking circuits. Errors, on the other hand, are detected by error detection mechanisms, such as acceptance tests, capability checks and time-outs. Upon detection of an error, a fault location procedure is called for to identify the faulty module. When a fault is detected by self-checking circuit, a damage assessment carried out separately in the following three cases [4,5]:

Case 1: An error is detected and the fault module is identified.

Case 2: An error is detected without the fault module.

Case 3: A fault is detected be periodic diagnosis.

The accuracy of damage assessment depends on the information collected from detection and diagnosis mechanism, which includes the location of the faulty module and error syndrome S, expressed as S= [v_{b1},t_1 ;.....;v_{bs}, t_s] where v_{b1}..... v_{bs} are the modules which have detected error(s),and t_1 .... t_s are times at which the respective modules detected there first error.

1.3 Rollback Recovery
Memory and persistent checkpoints are established periodically, the period for persistent checkpoints obviously being much larger. When a transient or a persistent failure is detected, the system is rolled back to the last checkpoint whether it is a memory or a persistent checkpoint and the treatment slightly differs. When a power cut occurs, the system is rolled back to the last persistent checkpoint [6].

1.3.1 Handling transient failures
Upon detection of a transient failure, if the last checkpoint is a memory checkpoint, all dirty active pages present in memory are discarded, pure recovery pages are restored into readable recovery data whereas readable recovery data as well as clean copies remain unchanged [7]. If the last checkpoint is a persistent one, memory pages (recovery or dirty active data) are discarded, since the recovery data is the data stored on disk.

1.3.2 Handling permanent faults

Upon detection of a permanent fault, the same rollback procedure is applied as in the case of a transient failure. Nevertheless, in addition, the faulty node has been lost as well as its disk and memory contents [8].

2. CHECKPOINTING

A checkpoint is a local state of a process saved on the stable storage. In a distributed system, since the processes in the system do not share memory, a global state of the system is defined as a set of local states, one from each process. The state of channels corresponding to a global state is the set of messages sent but not yet received. A global state is said to be "consistent" if it contains no orphan message [9]. A global checkpoint consists of a set of local checkpoints, one for each process, from which a distributed computation can be restarted after a failure. A local checkpoint is a state of a process saved onto stable storage [10].

Checkpointing is the process of saving the status information. Coordinated checkpointing is an attractive approach for transparently adding fault tolerance to distributed applications since it avoids domino effects and minimizes the stable storage requirement. A checkpoint is a local state of a process saved on stable storage [11]. Local checkpoint is an event that records the state of a process at processor at a given instance. Checkpoint may be local or global depending on taking the checkpoints and Local checkpoint is an event that records the state of a process at processor at a given instance. Checkpoint may be local or global depending on taking the checkpoints [12].

2.1 Related works on Check-pointing algorithms

Chandy and Lamport proposed a global snapshot algorithm for distributed systems. It is observed that every check-pointing algorithm proposed for message passing system uses Chandy and Lamport's algorithm as the base. Ravi Prakash and Mukesh Singhal had described a Synchronous Snapshot collection algorithm for Mobile Systems that neither forces every node to take a local snapshot nor blocks the underlying computation during snapshot collection. Guohong Cao and Mukesh Singhal had proposed an efficient algorithm that neither forces all the processes to take checkpoints nor blocks the underlying computation during checkpointing and which significantly reduces the number of checkpoints.

Guohong Cao and Mukesh Singhal had introduced the concept of "Mutable Checkpoint" which is neither a tentative checkpoint nor a permanent checkpoint to design efficient check-pointing algorithms for mobile computing system. Mutable Checkpoint can be saved anywhere e.g. the main memory or local disk.

A consistent global checkpoint is a set of states in which no message is recorded as received in one process and as not yet sent in another process. J. L. Kim and T .Park had presented a new efficient synchronized check-pointing protocol which exploits the dependency relation between processes in distributed systems. Yanping-Changhui-Yandong presented a protocol which is integration of two approaches: time based and index-based. In present description an index-based check-pointing protocol has been developed, which uses time to indirectly coordinate the creation of consistent global checkpoint for mobile computing systems [13].

Weigang Ni Susan V Vrbsky and Sibabrata Ray had presented a new checkpoint algorithm for mobile distributed system. Ravi Prakash and Mukesh Singhal had presented a global snapshot algorithm that combines the information collected by the initiator. This generates a maximal, consistent global snapshot that is more recent than the snapshot collected by any initiator. A maximal snapshot implies that the amount of computation lost during rollback after node failures is minimized [14,15].

Bidyut Gupta Shahram Rahimi and Ziping Liu had presented a non-blocking coordinated Check-pointing algorithm suitable for mobile environments. Ch. D. V. Subba Rao and M.M. Naidu had proposed a new checkpointing protocol combined with selective sender based message logging .The protocol is free from the problem of lost messages. This protocol minimizes different overheads i.e. checkpointing overhead, recovery overhead, blocking overhead. Nuno Neves and W. Kent Fuchs had proposed a new check-pointing protocol for distributed systems, was designed to take into consideration the special characteristics of mobile environments.

Kanmani, Anitha and Ganesan proposed a new approach which is used to reduce the much overheads of the previous non-blocking algorithms. The new algorithm is based on the timeouts of coordinator process. Instead of storing a single checkpoint like other non-blocking algorithms, it sets three checkpoints. Kumar, Mishra and Joshi presented a non-blocking minimum process coordinated check-pointing protocol that not only minimizes useless checkpoints but also minimizes overall bandwidth required over wireless channels.

3. Z-PATHS AND Z-CYCLES

In a distributed computation, each ordered pair of processes is connected by an asynchronous, reliable, directed logical channel with unpredictable but finite transmission delays, i.e. no message will be lost in the channel. Moreover, a process can execute internal, send and receive statements, where the last two are represented as "send(m)" and "receive(m)" in the text, respectively. Processes of a distributed computation are sequential: each process produces a sequence of events. All the events produced by a distributed computation can be partially ordered with Lamport's well-known happened-before relation (Lamport, 1978), denoted as "→hb " in the text.

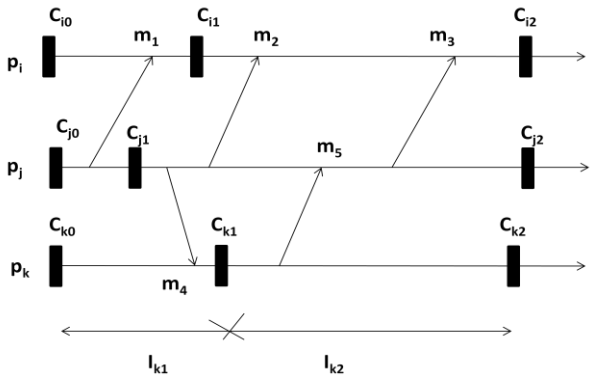

Figure 1: An example checkpoint and communication pattern

A distributed computation is usually represented as a checkpoint and communication pattern, like the one depicted in Fig. 1. $C_{i,x}$ represents the xth checkpoint of process P_i. The sequence of events occurring at P_i between $C_{i,x-1}$ and $C_{i,x}$ ($x > 0$) is called an interval, denoted by $I_{i,x}$. Furthermore, each process Pi starts its execution with an initial checkpoint $C_{i,0}$.

Netzer and Xu (1995) introduced the following notions of Z-paths and Z-cycles.

Definition 1. A Z-path is a message chain such that every message in it is received in the same or an earlier interval than the succeeding message is sent. In addition, if a Z-path satisfies that its first message is sent by P_i after checkpoint $C_{i,x}$ and its last message is received by P_i before checkpoint $C_{i,y}$, we say that this Z-path is from $C_{i,x}$ to $C_{i,y}$. For example, in Fig. 1, message chain [m_5, m_2] constitutes a Z-path from $C_{k,1}$ to $C_{k,2}$, but another message chain [m_5, m_1] is not a Z-path. A Z-path from a checkpoint $C_{i,x}$ to the same one is called a Z-cycle. For instance, Z-path [m_5, m_4] is a Z-cycle [13].

4. MODELING ERROR PROPAGATION

Errors in one module can propagate to other module via propagation paths, which are communication paths from the source to the destination with distinct intermediate modules [16]. The error propagation time from v_i to v_j. Denoted by X_{ij} is defined as the time interval between the contamination time of v_i to v_j. The error propagation times will be derived from B_{ij}'s, which is defined as the time from an error to propagate from v_i to one of its neighbors, v_j, via different communication channel between them. B_{ij}'s are assumed to be independent of each other. Then X_{ij} is the minimum propagation time among all the paths from v_i to v_j represented by graph from figure 2 is

X_{13}= min ($B_{12} + B_{23}$, $B_{14} + B_{45} + B_{53}$, $B_{14} + B_{45} + B_{53} + B_{23}$, $B_{12} + B_{24} + B_{45} + B_{53}$).

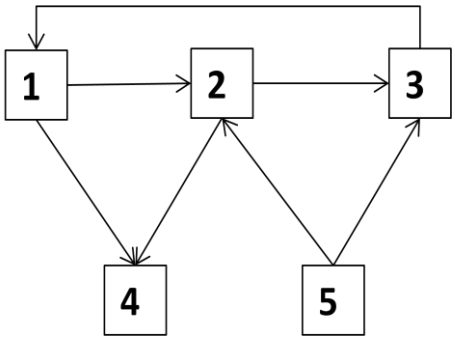

Figure 2: System graph, 1, 2, 3, 4, and 5 are nodes of the system

A single level fault detection mechanism has the property that faults are detected immediately upon their occurrence. If a fault is detected in a module during periodic diagnostics, errors might already have been induced and propagated to other modules. An undetectable fault can be captured only during the fault diagnosis. After the errors induced by this fault are detected by some detection mechanisms. The probabilities of any fault to be FD-detectable and PD-detectable, denoted by C_i^F and C_i^P respectively, are assumed to be fixed and known [17].

5. DAMAGE ASSESSMENT MODULES

5.1 Module 1

An error is detected and the faulty module is identifies.

In this module both S and module v_k are known. Damage assessment will start from v_k , the source of errors. To derive δ_k (t) it is essential to calculate

1) f_k^T ,the density function of T_k^C without considering the syndrome,

2) the error syndrome conditional likelihood $L(S,\tau)$ which is conditional probability of S given that v_k is the faulty module and $T_k^C = \tau$.

A complete diagnosis is assumed to have 100% coverage so that v_k should be fault free immediately after T_k^D , which can be as early as T_k^Y ,the last faulty time. Therefore, the density function of T_k^F is expressed as

If $T_k^D <= t < T_k^P$, then

$$f_k^{T^F}(t) = \frac{(1-C_k^F - C_k^P)}{w_k} f_k^Y(t - T_k^Y)$$

Where T_k^P and T_k^D are time of v_k 's last diagnosis and periodic diagnosis. And if $T_k^Y <= t < t_1$, then

$$f_k^{T^F}(t) = \frac{(1-C_k^F)}{w_k} f_k^Y(t - T_k^Y)$$

Where f_k^Y (.) is the density function of Y_k and W_k is the normalizing constant.

Define E_{ij}, the error latency from v_k to v_j ,is the time interval from the v_k 's contamination time to the time v_j 's detection time [19,20] .i.e.,

$$E_{kj} = X_{kj} + K_j$$

5.1.1 Case 1

If j=k, then $E_{kj} = K_j$,propagation of errors from a faulty module v_k into other, modules is characterized by the joint distribution of E_{k1} ,E_{k2},.....,E_{kn} and ,thus , the error syndrome's conditional likelihood can be calculated as

$$L_i(S,\tau) = \text{Prob}[\, E:ib_1 = t - \tau \,, \ldots\ldots E:ib_s = t - \tau \,, \; E:iw_1 > T - \tau \,, \ldots\ldots E:iw_{n-s} > T - \tau \,]$$

Where T is the current time instance at which damage assessment is done. Using Baye's equation δ_k (t) is derived as,

$$\delta_k(\mathrm{t}) = \int_{\mathrm{T_k^D}}^{\infty} \mathrm{T_j^C} \frac{L_k((S,\tau) f_k^{T^C}(t)}{\int_{\mathrm{T_k^D}}^{t} L_k((S,\tau) f_k^{T^F}(\tau)} d\tau$$

5.1.2 Case 2

If $j \neq k$, to estimate the damage on module v_i, first calculate $L_{kj}(S, T_k^C, T_j^C)$ the likelihood of S. Obviously

$L_{kj}(S, T_k^C, T_j^C) = 0$ if $\min(t_1, T_j^C) < T_k^D$, and

$$L_{kj}(S, T_k^C, T_j^C) = \mathrm{Prob}[E:ib_1 = t-\tau, \ldots\ldots E:ib_s = t-\tau, E\,iw_1 > T-\tau, \ldots E\,iw_{n-s} > T-\tau]$$

Where $E_{kj}(s) = X_{kj}(s) + K_i$, and $X_{kj}(s)$ is the error propagation time from v_k to v_i under the condition that $X_{kj} = s$. In other words $X_{kj}(s)$ can be viewed special case of X_{kj} with two sources of errors. One source is v_k, from which errors propagate to v_i via all possible path between v_k and v_i except for those passing through v_i .The other source is v_j, which starts the propagation of error to v_i via all possible paths between v_j and v_k at s time units after v_k become faulty. This interpretation simplifies the evaluation of $X_{ijk}(s)$.

From figure 1 $X_{134}(s)$ can be evaluated as

$X_{134}(s) = \min\{ B_{12} + B_{23}, s + X_{43} \} = \min\{ B_{12} + B_{23}, s + B_{45} + B_{53}, s + B_{45} + B_{52} + B_{23} \}$

With the knowledge of $L_{kj}(S, T_k^C, T_j^C)$ for all T_k^C and T_j^C,

$$\delta_j(t) = \frac{\int_{T_k^D}^{t} L_k(S, T_k^D, t) f_k^{T^F}(T_k^C) dT_k^C}{\int_{T_k^D}^{\infty}\int_{T_k^D}^{t} L_k(S, T_k^C, t) dT_k^C d\tau}$$

$$= \frac{\int_{T_k^D}^{t} L_{kj}(S, \tau, t) f_k^{T^C}(\tau_k) d\tau_k}{\int_{T_k^D}^{t} L_k(S, \tau_k) f_k^{T^C}(\tau_k) d\tau_k}$$

5.2 Module 2

In this module damage assessment must be made without any knowledge on the location of the faulty module. One example, when the fault diagnosis routine fails to locate the faulty module due to either the occurrence of a transient fault or insufficient coverage of the diagnosis routine [18].

Let π_i represent v_i's faulty probability without considering the error syndrome. These π_i''s are basic objective in locating the faulty module.If damage assessment is performed detection of an error, π_i is the same as the prior faulty probability π_i' determine by

$$\pi_i' = \frac{F_i^Y(T - T_i^Y) \prod_{j=1, j\neq i}^{N} (1 - F_i^Y(T - T_i^Y))}{\sum_{j=1}^{N} (F_j^Y(T - T_i^Y) \prod_{j=1, j\neq i}^{N} (1 - F_i^Y(T - T_i^Y))}$$

Where $F_i^Y(T - T_i^Y) \prod_{j=1, j\neq i}^{N} (1 - F_i^Y(T - T_i^Y))$ is relative suspicion of v_i being the faulty module.

5.3 Module 3

Finally, this module arises when a periodic diagnosis detect a fault in v_k. It implies that the fault may have occurred any time between the present and the previous periodic diagnostic completed at T_k^P. If the fault had occurred before T_k^P, it would have been detected by the previous periodic diagnostic. Using the Baye's equation, the distribution of v_k's faulty time in this module is derived as [21]

$$f_k^{T^F}(t) = \frac{f_k^T(t - T_k^Y)(1 - F_k^Y(T - t))}{\int_{T_k^P}^{T} f_k^T(\tau - T_k^Y)(1 - F_k^Y(T - \tau)) d\tau}$$

for $T_k^P <= t <= T$, Where T is the fault detection time.If Y_k is exponentially distributed, T_k^F can be calculated with the help of uniform distribution over the interval $[T_k^P, T]$.

6. OPTIMAL ROLLBACK POINT

Let T denote the current time. For the convenience of problem formulation is designated as the origin of the time axis and all events are enumerated backward from T. For example, the kth checkpoint means the kth previous checkpoint from T. Let cp_i^k be the time v_i established the kth checkpoint, and r_{jk} be the time v_i received the kth message. If v_i rolls back to the kth checkpoint, then v_i's rollback distance is $d_i(k) = t - cp_i^k$ The probability that v_i can successfully rollback to the kth checkpoint is denoted by $p_i(k)$. Hence

$$p_i(k) = \int_{p_i^k}^{\infty} \delta_i(\tau) d\tau = 1 - \Delta_i(cp_i^k).$$

The above equation holds only when vi is faulty module. If v_i is not faulty module its contamination is caused by incoming message from other module that is., error propagation. Since v_i receive messages only at r_{in}, $n >= 1$, if v_i is not the faulty module and $r_{in} < cp_i^P < r_{in} + 1$, then

$$p_i(k) = 1 - \Delta_i(r_i^n).$$

7. ROLLBACK RECOVERY

Rollback recovery is the sum of rollback distances (d_i's) in all modules. The main objective to minimize the mean recovery overhead denoted by O. Let Z denote the total overhead for a global restart which is invoked at the failure of rollback recovery. Rollback recovery fails if any module of the system rolls back to an incorrect checkpoint. Thus, the

probability of successful rollback recovery is the product of the probability that each module's rollback point is correct. Hence O can be expressed as [22, 23].

$$O = (\prod_{i=1}^{N} P_i(k_i)) \sum_{i=1}^{N} d_i(k_i) + (1 - \prod_{i=1}^{N} P_i(k_i))(\sum_{i=1}^{N} d_i(k_i) + Z)$$
$$= \sum_{i=1}^{N} d_i(k_i) + (1 - \prod_{i=1}^{N} p_i(k_i))Z$$

Where ki is a nonnegative integer indicating the rollback point of v_i. There is an upper limit C_i, for $k_{i,1}$<=i<=N, because each secure storage will have a limited capacity. Then the optimal rollback problem can be represented as

Minimize

$$\sum_{i=1}^{N} d_i(k_i) + 1 - \prod_{i=1}^{N} p_i(k_i)Z$$

Subject to constraint

$k_i <= C_i$ for 1<=i<=N.

This is a non-linear integer programming problem. To develop an algorithm for a non-linear integer programming problem the objective function O should follow the convexity property. If p_i is expressed as in $p_i(k)$,O can be viewed as a continuous function of d_i's. The following lemma provides the necessary and sufficient condition for the complexity of O which is provided in [24].

Lemma 1

O is convex with respect to d_i iff $\delta_i(t), t = T - d,$ is monotonically increasing.

7.1 ROLLBACK RECOVERY ALGORITHM

Step 1.

For all i= 1(2) N, Assign k_i be the smallest integer such that

$$cp_i^{k_i} \leq T_i ,$$

Let k_i:=C_i and if $k_i > C_i$ then $Y := (\prod_{i=1}^{N} p_i(k_i))Z$.

Step 2.

Let S:= {i : k_i <C_i }.If ki:=Ci ,for all i ∈ S then go to step 6.

Step 3.

For all i ∈ S, $y_i := \delta d_i - (\rho_i - 1)Y$

Where

$$\delta d_i := d_i(k+1) - d_i(k_i), \rho_i := 1 + \frac{\delta p_i}{p_i(k_i)},$$
$$\delta p_i := p_i(k_i + 1) - p_i(k_i)$$

Step 4.

If yi >=0 for all ii ∈ S , then go to step 5.

Otherwise, for all i ∈ S and yi < 0, then

k_i:= k_i+1 and $Y := \rho_i Y$

Go to steo 2

Step 5.

For any $A \subset S$, then

$$D(A) := \sum_{i \in A} \delta d_i - (\prod_{i \in A} \rho_i - 1)Y$$

Find a set $S^* \subset S$ such that D(S*) < 0.

If S* does not exist, then go to step 6 . Otherwise, for all

i ∈ S*, then k_i:= k_i+1 and $Y := \rho_i Y$.

Go to step 2

Step 6.

Terminate the algorithm and the current value of k_i, for all i:=1(2)N, is the optimal recovery point for v_i.

Rollback Recovery is essentially an algorithm which searches from smaller k_i's towards larger k_i's. In step 1 k_i's is initialized to the smallest value that will put the corresponding cpik inside the convex region of the function O with respect to k_i's 's,by Lemma 1,the optimal solution for k_i's is obtained when O cannot be reduced any further by incrementing any subset of k_i's. If k_i reaches its limit C_i before the minimum O is reached,the optimal solution for k_i will be C_i .If all k_i's reach their limits during the search, Rollback Recovery algorithm terminates immediately(step 2).

The search of the above algorithm conducted in two levels. In first level (step 3 and step 4),check whether incrementing a legitimate k_i's , would result in a smaller mean overhead. The set S defined in step 2 is the set of all legitimate k_i's at that moment. The variable yi is actually the difference in O with ki and k_i+1, that is , Y_i=O (.....ki+1,.....) - O(.....k_i .....), if y_i'>=0 ,then next level of search is performed in step 5.If at least one yi is negative, all k_i 's with negative y_i will be

incrementing by one, which yield an even smaller overhead because of the following lemma 2[25].

Lemma 2

Let $y_{i,j}=1(2)n$ be defined as in step 3 of Rollback Recovery algorithm. If $y_j<0$ and $y_{i,0}$, then $O(.....k_i+1...)<O(...k_i...)$.

8. INTEGRATION OF DAMAGE ASSESSMENT

Many rollback recovery schemes have been proposed for distributed nbut very few of them have been considered corrupted checkpoints and damage assessment. To integrate damage assessment with existing schemes, especially the indexed-based check-pointing algorithm such as one proposed by BQF(Baldoni, Quaglia, Fornara) [3] .In general, any rollback recovery scheme equipped with damage assessment should comply with following steps after detecting error [26].

8.1 Damage assessment

Using the algorithms for determining the optimal rollback points, compute a set of checkpoints which are safe to rollback, while considering the overall overheads [27].

8.2 Cancellation of corrupted messages

All checkpoints established after the set of safe checkpoints are consider to have been corrupted. Hence, all messages sent after a corrupted checkpoint are also corrupted and should be removed from the receivers' stable storage if they have been logged. The messages sent between a safe checkpoints and its next checkpoint and considered correct if they have not been cancelled. In some rare event, rollback may propagate from a saved checkpoint if one of the messages received before the establishment of the safe checkpoint has been cancelled [28].

9. SEARCH FOR A RECOVERABLE SYSTEM STATE

A system will be a recoverable state after cancelling all corrupted messages, since most messages would have been logged if they can survive the cancellation. If not, a new recoverable system state must be determined from the current state [29].

9.1 Recovery

Rollback to the derived recoverable system state will recover successfully if the system sate is indeed not corrupted is predicted in the damage assessment [30].

The main difficulty of integrating damage assessment into an existing rollback recovery scheme is the size of the stable storage. Without considering error propagation, whenever a checkpoint becomes a part of recoverable system state, all messages received before the checkpoint can be safely discarded from the stable storage. With damage assessment, however, checkpoints and messages must be kept in the stable storage for a longer period, thus requiring a large stable storage [31].This is more expensive time and resource mechanism to detect error to enhance the coverage remove the need for damage assessment.

10. CONCLUSION

Check-pointing for a rollback recovery in a current process have received considerable attention. Two crucial problems which must be resolved are the problem of rollback propagation (domino effect) and error propagation. In this paper discussed a method of damage assessment to handle the error propagation problem based on the earlier work on error propagation and fault location. The main important concept is that ,the equivalence classes between checkpoints provides a framework that can be used to design efficient checkpoint time stamping mechanism. Such mechanism can be used in any checkpoint algorithm to reduce sequence numbers of the increasing process.

11. FUTURE SCOPE

The damage assessment model developed in this paper can also be used to solve other problems, such as the scheduling of periodic diagnostics and the determination of optimal inter check-point intervals. These problems are matter of future inquiry.

12. REFERENCES

[1] R. Koo and S. Toueg, "Check-pointing and Rollback-Recovery for Distributed Systems", IEEE trans. Software Engineering, vol. SE-13, no. 1, pp. 23-31, Jan 1987.

[2] Chandy K. M. and Lamport L., "Distributed Snapshots: Determining Global State of Systems," ACM Transaction on Computing Systems, vol. 3, No. 1, pp. 63-75, February 1985.

[3] R. Baldoni, J. Brezinsky, J.M. Helary, A. Mostefaoui and M. Raynal, On Modeling Consistent Checkpoints and the Domino Effect in Distributed Systems, Proc. IEEE Int. Conference on Future Trends in Distributed Computing Systems,1995, pp.314–323.

[4] Pradhan D.K., Krishana P.P. and Vaidya N.H., "Recovery in Mobile Wireless Environment: Design and Trade-off Analysis,"Proceedings 26th International Symposium on Fault-Tolerant Computing, pp. 16-25, 1996.

[5] E.N. Elnozahy, D.B. Johnson and Y.M. Wang, A Survey of Rollback-Recovery Protocols in Message-Passing Systems,Technical Report No.CMU-CS-96-181, School of Computer Science, Carnegie Mellon University, 1996.

[6] Y.M. Wang, Consistent Global Checkpoints that Contains a Set of Local Checkpoints, IEEE Transactions on Computers,vol. 46, no. 4, 1997, pp. 456-468.

[7] Proceedings of International Conference on Parallel Processing, pp. 37-44, August 1998.

[8] R. Baldoni, J. Brezinsky, J.M. Helary, "A. Mostefaoui and M. Raynal, On Modeling Consistent Checkpoints and the Domino Effect in Distributed Systems", Proc. IEEE Int. Conference on Future Trends in Distributed Computing Systems,1995, pp.314–323.

[9] E.N. Elnozahy, D.B. Johnson and Y.M. Wang, "A Survey of Rollback-Recovery Protocols in Message-Passing Systems", Technical Report No.CMU-CS-96-

181, School of Computer Science, Carnegie Mellon University, 1996.

[10] Y.M. Wang, "Consistent Global Checkpoints that Contains a Set of Local Checkpoints", IEEE Transactions on Computers,vol. 46, no. 4, 1997, pp. 456-468.

[11] W. Wagealla, T. Osman, and A. Bargiela, "Error detection algorithm for agent-based distributed applications," in Proc. 2nd Workshop Agent-Based Simulation, Passau, Germany, 2001, pp. 106–110.

[12] G. Cao and M. Singhal,"Mutable Checkpoints:A New Check-pointing Approach for Mobile Computing Systems", IEEE Transactions On Parallel And D istributed Systems,Vol.12,No.2,February 2001,pp 157-172.

[13] Tsai, J., and Lin, J. W.,"On the Fully-Informed Communication-Induced Check pointing Protocol, "Proceedings of 11th Pacific Rim International Symposium on Dependable Computing, Changsha, Hunan, PRC, 2005,.

[14] R. Prakash and M. Singhal. "Low-Cost Check-pointing and Failure Recovery in Mobile Computing Systems". IEEE Trans. on Parallel and Distributed System, pages 1035-1048,Oct. 1996.

[15] Elnozahy E.N., Alvisi L., Wang Y.M. and Johnson D.B., "A Survey of Rollback-Recovery Protocols in Message-Passing Systems,"ACM Computing Surveys, vol. 34, no. 3, pp. 375-408, 2002.

[16] G. Cao and M. Singhal. "On impossibility of Min-Process and Non-Blocking Check-pointing and An Efficient Check-pointing algorithm for mobile computing Systems". OSU Technical Report #OSU-CISRC-9/97-TR44, 1997.

[17] Bidyut Gupta, S.Rahimi and Z.Lui. "A New High Performance Check-pointing Approach for Mobile Computing Systems". IJCSNS International Journal of Computer Science and Network Security, Vol.6 No.5B, May 2006.

[18] Acharya A. and Badrinath B. R., "Check-pointing Distributed Applications on Mobile Computers," Proceedings of the 3rd International Conference on Parallel and Distributed Information Systems, pp. 73-80, September 1994.

[19] Nuno Neves and W. Kent Fuchs. "Adaptive Recovery for Mobile Environments", in Proc .IEEE High-Assurance Systems Engineering Workshop, October 21-22, 1996, pp.134-141.

[20] Y. Manable. "A Distributed Consistent Global Checkpoint Algorithm with minimum number of Checkpoints". Technical Report of IEICE, COMP97-6(April1997)

[21] J.L.Kim and T.Park. "An efficient protocol for Check-pointing recovery in Distributed Systems" IEEE Transaction On Parallel and Distributed Systems,4(8):pp.955-960, Aug 1993.

[22] Yanping Gao, Changhui Deng, Yandong Che. "An Adaptive Index-Based Algorithm using Time-Coordination in Mobile Computing".International Symposiums on Information Processing, 2008.

[23] Kanmani - Anitha - Ganesan."Coordinated Check-pointing with Avalanche Avoidance for Distributed Mobile Computing System."International Conference on Computational Intelligence and Multimedia Applications 2007.

[24] W. Wagealla, T. Osman, and A. Bargiela,"Error detection algorithm for agent-based distributed applications," in Proc. 2nd Workshop Agent-Based Simulation, Passau, Germany, 2001, pp. 106–110.

[25] Sani Tripathy and Brajendra Panda," Post-Intrusion Recovery Using Data Dependency Approach ",Proceedings of IEEE Workshop on Information Assurance and Security United States Military Academy, West Point, NY, 5-6 June, 2001.

[26] B. Gupta, S.K. Banerjee and B. Liu, "Design of new roll-forward recovery approach for distributed systems", IEE Proc. Computers and Digital Techniques, Volume 149, Issue 3, pp. 105-112,May 2002.

[27] D. Manivannan, and M. Singhal, "Asynchronous recovery without using vector timestamps", Journal of Parallel and Distributed Computing, Volume 62,Issue 62 pp. 1695-1728, Dec 2002.

[28] M. Ohara., M. Arai., S. Fukumoto., and K. Iwasaki.,"Finding a Recovery Line in Uncoordinated Check-pointing", Proceedings 24th International Conference on Distributed Computing Systems Workshops (ICDCSW'04), pp. 628 – 633, 2004.

[29] B. Gupta, Y. Yang, S. Rahimi, and A. Vemuri, "A High-Performance Recovery Algorithm for Distributed Systems", Proc. 21st International Conference on Computers and Their Applications, pp. 283-288, Seattle, March 2006.

[30] S. Monnet, C. Morin, R. Badrinath, "Hybrid Check-pointing for Parallel Applications in cluster Federations", Proc. 4th IEEE/ACM International Symposium on Cluster Computing and the Grid, Chicago, IL, USA, pp. 773-782, April 2004.

[31] J. Cao, Y. Chen, K. Zhang and Y. He,"Check-pointing in Hybrid Distributred Systems",Proc.7th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'04), pp. 136-141, May 2004.