4N－1

時相論理によるマルチプロセッサシステムのロールバック手法について

米田 清光, 松原 隆, 古賀 義亮
防衛大学校 情報工学教室

1 まえがき

システムを高信頼化する技術として，ロールバック手法が知られている．本手法をマルチプロセッサシステムに適用し，各プロセッサが各々独立にチェックポイントを設定した場合，システム全体として一貫性のある状態から再実行を行うため，ドミノ効果[1]を生じる可能性がある．本稿では，時相論理を適用し，故障が生じた場合に迅速にロールバックを行うため，システム全体で一貫性があり，かつメッセージの喪失を生じないチェックポイントを求める手法を提案する．また，本アルゴリズムを適用して，ロールバック範囲を一定範囲内に抑えることができる．

2 諸定義

2.1 マルチプロセッサシステム

本稿では，n個のプロセッサ$\{P_0, P_1, \ldots, P_{n-1}\}$からなるマルチプロセッサシステムを仮定する．各プロセッサは共有メモリを持たず，その間の通信はメッセージの送受信のみにより行うものとする．また，プロセッサ間における通信には，高い信頼性のある通信技術が他の手法により実装されているものとする．

2.2 一貫性のあるチェックポイント

プロセッサの状態は，イベントによる状態の遷移で考えることができ，メッセージの送受信イベントによっても状態遷移が生じる．一貫性のない状態とは，あるメッセージが送信されていないにもかかわらず，そのメッセージが受信されている状態である[2]．ロールバック手法では，故障が生じていない正常動作時に逐次内部状態を保存し（チェックポイントと呼ばれる），故障が発生した場合には，このチェックポイントに戻り再実行を行うことにより故障から回復する．このため，システム全体として一貫性のあるチェックポイントから再実行を行う必要がある．システム全体の一貫性あるチェックポイントとは，各プロセッサP_iのチェックポイントの集合を$\{c_0, c_1, \ldots, c_{n-1}\}$とすると，すべてのチェックポイント組$(c_i, c_j)$において，$c_i$を設定した後に送信され，$c_j$を設定する前に受信されているメッセージ$m$が存在しないチェックポイントの集合である．しかし，一貫性のあるチェックポイントにロールバックしただけでは，メッセージの喪失を生じる可能性がある．

2.3 チェックポイント間隔及び依存関係

チェックポイント$c_{i,x}$と次のチェックポイント$c_{i,x+1}$の間を'チェックポイント間隔'といい，'$I_{i,x}$'と表す．また，チェックポイント間隔$I_{i,x}$と$I_{j,y}$間で通信が行われた場合，'チェックポイント$c_{i,x}$と$c_{j,y}$の間に依存関係がある'と呼び，'$c_{i,x}Dc_{j,y}$'と表す．この依存関係はP_iで生じた時点で，P_jでも同様に$c_{j,y}Dc_{i,x}$が生じるものとする．この依存関係には，以下に示す性質がある．

性質1：　$c_{i,x}Dc_{j,y} \wedge c_{j,y}Dc_{k,z} \Rightarrow c_{i,x}Dc_{k,z}$　(1)

この依存関係$c_{i,x}Dc_{j,y}$は，$I_{i,x}$，$I_{j,y}$のどちらか一方で故障が生じた場合，$c_{i,x}$，$c_{j,y}$にロールバックしなければならないことを示す．

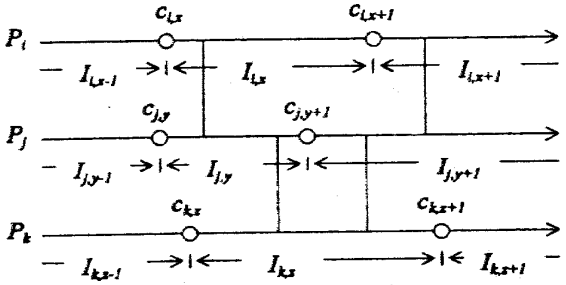

図 1: ロールバックポイント

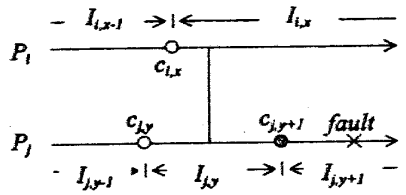

図 2: 保持条件

3 時相論理を用いた評価手法の提案

3.1 ロールバックポイントの確定法

各通信イベントが生じるごとに，性質1を用いて各プロセッサにおいて依存関係を求めることにより，ロールバックポイントを得ることができる．図1に一例を示す．図において依存関係を求めると，P_iの$c_{i,x}$に関する依存関係は，$c_{i,x}Dc_{i,x+1}$，$c_{i,x}Dc_{j,y+1}$となる．これらの依存関係からP_iは，P_i，或いはP_jで故障が生じた場合のロールバックポイント$c_{i,x}$を得ることができる．しかし，以前に設定されたチェックポイント及び生じた依存関係を保持しなければならず，多くのメモリを消費するため，本手法では，保持する必要のない依存関係及びチェックポイントを削除する．次に，チェックポイント及び依存関係を保持しなければならない条件を示す．図2において，依存関係$c_{i,x}Dc_{j,y}$は，P_jで次の$c_{j,y+1}$が設定された場合，P_jで故障が生じてもP_iは，ロールバックする必要がなく，この依存関係が存在しないと考えられる．また，$c_{i,x}$は，依存関係が存在しない場合，次の$c_{i,x+1}$が設定された時点で任意のプロセッサで故障が生じても$c_{i,x}$にロールバックすることはなく，保持する必要はない．依存関係がある場合，その依存関係が存在する限り保持しなければならない．従って，以下に示す保持条件が得られる．

(a) 依存関係$c_{i,x}Dc_{j,y}$は，次のチェックポイント$c_{j,y+1}$が設定されない限り保持しなければならない．

(b) $c_{i,x}$は，依存関係が存在しない場合，次の$c_{i,x+1}$が設定されない限り保持しなければない．更に，$c_{i,x}$に関する依存関係$c_{i,x}Dc_{j,y}$が存在する場合，その依存関係が存在する限り保持しなければならない．

3.2 時相論理演算子の導入

時相論理では，$\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$などの演算子に加え時間系列を扱う演算子を用い，状態の遷移とともにその有効期間も含めて取り扱う．本稿では，各論理式は，T(真)，

On Rollback Method in Multi-Processor System by Using Temporal Logic
Kiyomitsu Yoneda, Takashi Matsubara and Yoshiaki Koga
National Defense Academy
1-10-20 Hashirimizu, Yokosuka 239, Japan

F(偽)の2値を取るものとし，ブール演算子$\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$をそれぞれ，論理積，論理和，否定，含意，同値の意味で用い，次に示す時相論理演算子$W(A,B)$を導入する．

$|W(A,B)|_t$演算子は，時刻tにおいて，"Bが真である時点からAが真である限りBは真のままである"ことを意味する．その真偽値を次式で定義する．

$$\begin{aligned} |W(A,B)|_t &= T \Leftrightarrow {}^{\exists}s < t(|B|_s = T \\ &\wedge \ {}^{\forall}u(s < u < t \rightarrow |A \wedge B|_u = T)) \\ &\wedge \ |A|_t = T \wedge |B|_t = T \end{aligned} \quad (2)$$

本論文では，有限時間内の各命題の変化が有限回であり，${}^{\forall}t(|A|_t = \lim_{\epsilon \to -0} |A|_{t+\epsilon})$が成立すると仮定すると，次に示す性質が成立する．

$$|W(A,C)|_t \vee |W(B,C)|_t \Leftrightarrow |W(A \vee B, C)|_t \quad (3)$$

次に，W演算子によるチェックポイント及び依存関係の保持条件の表現について示す．

3.3 W演算子による保持条件の評価

命題$Q(c_{i,x})$を"チェックポイント$c_{i,x}$が存在する"，$Q(c_{i,x}Dc_{j,y})$を"依存関係$c_{i,x}Dc_{j,y}$が存在する"と定義すると，次のように各保持条件を表現できる．

依存関係の保持条件より，

$$E_{c_{i,x}Dc_{j,y}} = W(\neg Q(c_{j,y+1}), Q(c_{i,x}Dc_{j,y})) \quad (4)$$

この論理式は，$E_{c_{i,x}Dc_{j,y}}$が真のとき$c_{i,x}Dc_{j,y}$を保持しなければならないことを意味し，偽のとき$c_{i,x}Dc_{j,y}$を削除できることを意味する．

チェックポイントの保持条件より，

依存関係が存在しない場合：

$$E_{c_{i,x}} = W(\neg Q(c_{i,x+1}), Q(c_{i,x})) \quad (5)$$

依存関係が存在する場合：

$$\begin{aligned} E_{c_{i,x}} &= W(\neg Q(c_{i,x+1}), Q(c_{i,x})) \\ &\vee W(Q(c_{i,x}Dc_{j,y}), Q(c_{i,x})) \end{aligned} \quad (6)$$

3.4 ロールバックポイント評価・確定アルゴリズム

次に，W演算子による現在存在するチェックポイント及び依存関係を求めるアルゴリズムを示す．

case1:P_iでチェックポイント$c_{i,x}$が設定された場合

(a) チェックポイントの保持条件を示す論理式を式(5)より生成する．

(b) 現在存在する論理式を評価し削除できる論理式を削除する．

(c) $c_{i,x}$の設定をすべてのプロセッサにブロードキャストする．

case2:P_iで依存関係$c_{i,x}Dc_{j,y}$が生じた場合

(a) $c_{i,x}Dc_{j,y}$の保持条件を示す論理式を式(4)より生成する．

(b) $c_{i,x}$の論理式を式(6)により変更する．

(c) $c_{i,x}$に関するすべての依存関係を他のすべてのプロセッサにブロードキャストする．

case3:依存関係発生に関するブロードキャストメッセージを受信した場合，現在存在する論理式を評価し，性質1より生じる依存関係の論理式を生成する．

case4:チェックポイント生成に関するブロードキャストメッセージを受信した場合，現在存在する論理式を評価し，削除できる論理式を削除する．

また，現在存在する最も古いチェックポイントに関する論理式が偽となるよう他のプロセッサにチェックポイントの設定要求を行うことにより，ロールバック範囲を縮小することができる．これは，上記のアルゴリズムの**case1**を次のように変更することで実現できる．

case1':P_iで新たな$c_{i,x}$が設定された場合，或いは新たなチェックポイント設定要求を受信した場合，

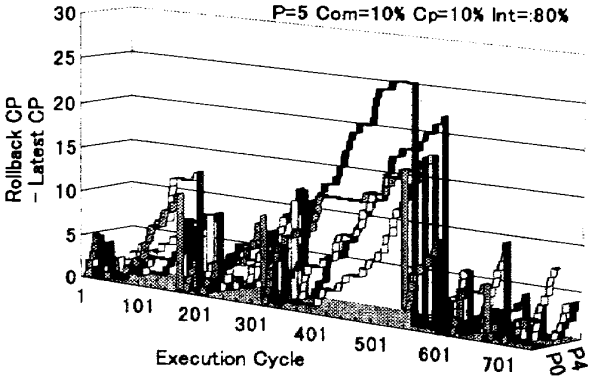

図 3: 適用例1

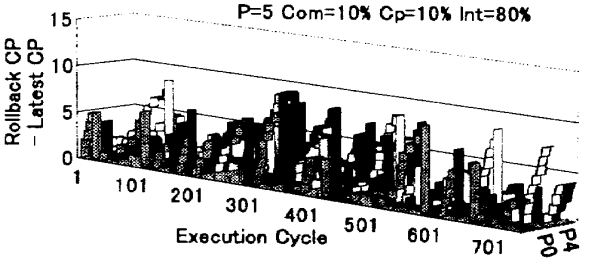

図 4: 適用例2

(a) チェックポイントの保持条件を示す論理式を生成する．

(b) 現在存在する論理式を評価し，削除できる論理式を削除する．

(c) 最新のチェックポイント番号と最古のチェックポイント番号の差が一定値を越えた場合，$c_{i,x}$の設定を通知するブロードキャストメッセージにチェックポイント設定要求を付加し送信する．

4 シミュレーションによる確認結果

各手法を単位実行サイクルあたりの通信の比率及びチェックポイント設定比率を各10%としてランダムに生成した動作パターンに適用し実験を行った．また図4では，チェックポイント範囲の制限を10とした．この結果より，各々のプロセッサでロールバックポイントが動的に得られること，及びロールバックをある範囲内に縮小できることが確認できた．

5 まとめ

本稿では，故障が生じた場合に，速やかにロールバックを行うため，各プロセッサにおいて，一貫性があり，メッセージの喪失を生じないロールバックポイントを各々独立に求める手法を提案した．これにより故障発生を検出したならば，直ちに各プロセッサが正確なロールバックポイントから再実行できるようになった．また，本手法を適用してロールバックをある範囲内に縮小できることが明らかにされた．

参考文献

[1] B.Randell, "System Structure for Software Fault Tolerance," IEEE Trans. Software Eng., Vol. SE-1, No. 2. pp. 220-232, June 1975

[2] K.M.Chandy and L.Lamport, "Distributed Snapshots:Determining Global States of Distributed Systems," ACM Trans. Comput. Syst., Vol. 3, No. 1, pp. 63-75, Feb. 1985

[3] D.M.Gabbay, I.Hodkinson and M.Reynolds, "Temporal Logic, Mathematical Foundations and Computational Aspects," Vol. 1, Oxford Science Publications, 1994