分散プロセス環境下におけるソフトウェア回復方式の一提案

2T－8

野村 立　倉持 和彦　齋藤 正史

三菱電機(株) 情報システム研究所

1　はじめに

分散システムの高信頼化の一方法として、あるプロセスが故障した場合に他のプロセスがその途中結果を引き継いで処理を続行する方法(Check Point Save/Roll Back)がある。

現在、OSの仮想記憶機能を利用したCheck Point Save機能については様々な方法が提案されているが、それらは変更ページをCheck Point発生時に保存する、というものであり、その処理の間該当プロセスは停止させておかなければならないという問題がある。実時間性を要求するアプリケーションにおいては連続停止時間の長さに制限を設ける場合があり、このような方法は使用できない。

またRoll Backについては、個々のプロセスが勝手にCheck Point Saveを実施するだけでは、メッセージが喪失しドミノ効果が発生する場合があるため、全プロセスのCheck Pointの組合せのうち、メッセージを喪失せず回復できる組合せ(Consistent Cut)を発見する必要がある。

ここでは、変更ページの保存処理を分散させCheck Point発生時に起こるプロセスの停止を最小限に抑えるCheck Point Save方法と、各プロセスのメッセージ送受信を監視して最新のConsistent Cutを発見する方法を提案する。

2　仮想記憶機能による Check Point Save 機能

2.1　構成

Check Point Save機能の構成を図1に示す。

ページ保存マップを世代別に複数個用意する。ページ保存マップには、各世代間で変更されたページの論理アドレスとそのデータを保存している物理ページアドレスを納める。

2.2　処理の概要

アドレス変換マップに対する処理は以下のようになる：

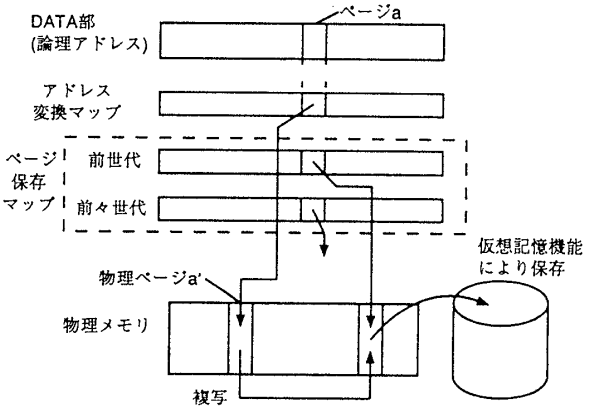

図 1: Check Point Save 機能の構成

1. 初期設定では全てのページをReadonlyモードに設定する。

2. あるページへのWrite処理が発生した場合：

   そのページがReadonlyモードの場合は、

   (a) 新しい空き物理ページを確保し、そこに当該ページのデータをコピーする。

   (b) 次に前世代アドレス変換マップの当該ページのエントリに新しく確保した物理ページをマップする。

   (c) 最後にそのページをRead/Writeモードに変更して、Write処理を続行する。

   そのページがRead/Writeモードの場合は、そのWrite処理を続行するのみ。

3. Check Pointに到達した場合には、新しいページ保存マップを確保し、それを最新世代の保存マップとする（各世代の保存マップが一つずつずれることになる）。

3　分散環境での Consistent Cut の発見

個々のプロセスが勝手にCheck Point Saveを実施していると、メッセージが喪失する可能性がある。例えば図2の様な場合で、それぞれのプロセスがCheck Point3

A Software Failure Recovery Technique for Distributed Processing Systems

Ritsu NOMURA, Kazuhiko KURAMOCHI, Masashi SAITO

Mitsubishi Electric Corp.

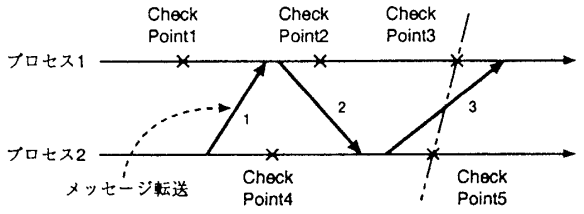

図 2: Check Point とメッセージ送信

と Check Point5 から回復しようとするとメッセージ 3 が消失してしまう。

従って Consistent Cut 発見のためにはメッセージ送受信のすき間に存在する Check Point を発見する必要がある。

3.1 Consistent Cut の発見方法

各プロセスで Check Point とメッセージ送受信のログを保存しておく。プロセス群の他にこの Check Point とメッセージ送受信のログを監視するプロセスをシステム内に常駐させる (図 3)。

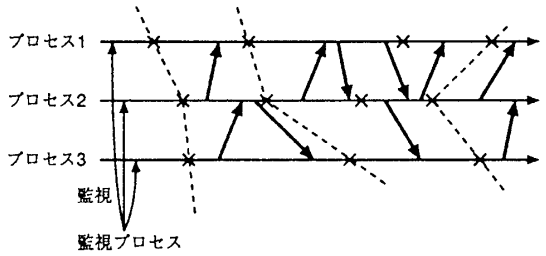

図 3: Consistent Cut の発見

監視プロセスは、それぞれのプロセスの Check Point を Consistent Cut の候補として選ぶ。

それぞれの Check Point について、それ以前に発生した全てのメッセージ送受信事象に対してその相手プロセスのメッセージ送受信事象を探索する。もし、相手プロセスでの事象が、相手プロセスの現在の Consistent Cut の候補となる Check Point より新しいものならば、相手プロセスの Consistent Cut 候補をそのメッセージ送受信事象の次の Check Point に変更する。

この処理を全プロセスに対して実施する。処理が停止した時点の Consistent Cut 候補の組合せが、新しい Consistent Cut になる。

3.2 停止性の保証

しかし、上記処理方式は停止する保証がない。特にメッセージが錯綜する場合などには Consistent Cut が得られない場合も考えられる。そこで Consistent Cut 処理の停止を保証するために、ある Consistent Cut から次の Consistent Cut を発見するまでに一定以上の時間を経過した場合、監視プロセスが全プロセスに、マーカメッセージを送信して、全プロセスのメッセージ送信を一時停止する必要がある。

4 考察

この方式をとることによって、Check Point Save に要する処理は新しい保存マップの確保のみとなり短時間で終了でき実業務の停止の短縮化が可能である。また、ページ保存マップの複数化によって複数世代のデータ保存が可能となる。

さらに、分散システム上での Roll Back 時にドミノ効果の発生を抑えるような Consistent Cut を効率良く発見できる。また、新しい Consistent Cut を発見した場合にはそれ以前の Check Point 情報は不要となる。従って各プロセスでのそのような Check Point 情報を削除すれば、計算機資源の節約に継る。

5 おわりに

今回提案した方式によって、システム実業務への負荷をできるだけ小さくした Check Point Save/Roll Back 機能が実現可能である。

現在、本方式に基いたシステムを試作中であり、試作完了後に本方式の評価を行なう予定である。

参考文献

[1] Chandy, K.M. and Lamport, L.: Distributed Snapshots:Determining Global States of Distributed Systems, ACM Trans. Computer System, (Feb. 1985)

[2] Storm,R.E. and Yemini,S.: Optimistic Recovery in Distributed Systems, ACM Trans. Computer System, (Aug. 1985)

[3] 真鍋、青柳: 分散チェックポイント・ロールバックアルゴリズム, 情報処理 (Nov.1993)