2P−9　分散システムにおける障害回復法に関する検討
—分散モニタシステムの一機能としての実装—

増岡 義政, 相田 仁, 齊藤 忠夫
東京大学 工学部

1　はじめに

計算機システムにおいて、障害が発生したときにアプリケーションを途中から回復する機能を持つことは、システムの信頼性を高める上で重要である。しかし疎結合マルチプロセッサシステムにおいては、各プロセッサの独立性が高いため、この障害回復機能を実現するのは簡単ではない。

本稿では、この分散システムの障害回復機能を、分散システムモニタの機能の1つとして実現する方法について検討する。また、ここで述べた方法をLANで接続された複数のUNIXワークステーションに実装し、アプリケーションの性能がどの程度影響されるかについてシミュレーションにより評価した結果を示す。

2　障害回復アルゴリズムとその実装上の課題

分散システムにおける障害回復の方法の1つとして、処理の途中でシステム全体の状態（各プロセスとチャネルの状態：以下システム状態と呼ぶ）を記録（チェックポインティング）し、障害が発生したときにそこへ戻って処理を再開（ロールバック・リカバリー）するというやり方がある。この方式では、システムは過去のシステム状態を保存しておかなくてはならないが、疎結合の分散システムではプロセス間の同期のためのクロックを使えないので、一貫性のある形でシステム状態を記録するのは容易ではない。

こうした問題点を克服して障害回復を実現するためのアルゴリズムが過去にいくつか提案されてきた[1][2]が、これら基本的なアルゴリズムを実際に計算機システムに実装するためには、まだ解決すべき課題がある。例えば：

- チャネルの状態（送信されたがまだ受信されていないメッセージ列）を記録するためには、送信側のプロセスは、自分がいつ局所状態を記録したかを受信側のプロセスに知らせなくてはならない。このため一般に各メッセージにヘッダを付けるというやり方がとられているが、この場合メッセージの境界が保存される必要があり、しかもメッセージが（ヘッダと共に）受信バッファに長時間残される場合があるなど、アプリケーションによって一回のチェックポインティングにかかる時間が不定となる。
- 計算機の信頼性が飛躍的に進歩している現在では、アプリケーションの通常時の（障害が発生しなかったときの）性能が、障害回復機能を実装したことによって影響を受けないようにすることも重要である。しかし1つのプロセスでアプリケーション本来の処理とチェックポインティングのための処理の両方を行うことは、スループットを大きく低下させる恐れがある。

これらの点を踏まえて、障害回復機能を実現する方法について検討した。次節以下ではその方法について説明する。

3　モニタシステムの一機能としての障害回復

当研究室では、研究テーマの1つとして、図1のようにアプリケーションの各プロセス(AP)を実行するプロセッサに、モニタのためのプロセス(LM)を実行するプロセッサを密結合させたモデルを用いて、このモニタシステムに様々なモニタリングのための機能を実現する方法を検討している。この分散システムモニタに障害回復機能を持たせようとする場合、上で述べた点に対して次のような利点があると考えられる。

- 図2のようにすることにより、各メッセージにヘッダを付ける必要がなくなり、また通信相手が局所状態を記録したことをすぐに知ることができる。そのため一回のチェックポインティングの時間が短縮され、またアプリケーションの構造にあまり依存しなくなる。
- プロセスがAPとLMに分離しているので、通常時はアプリケーションの動作はあまり影響されない。

また、障害回復機能に関する限り、LMが処理を実行するのはチェックポインティングおよび障害が発生してからの回復処理だけであるから、モニタの他の機能が影響を受けることは少ない。

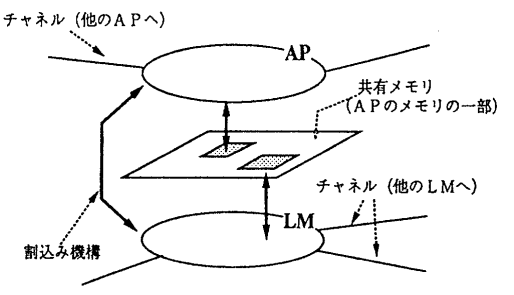

図 1: 分散システムモニタのモデル

"An Implementation of Checkpointing and Rollback Recovery for Distributed Computing Systems"
Yoshimasa Masuoka, Hitoshi Aida, Tadao Saitoh
The University of Tokyo

4 実装

以上を基本的な考え方として、図1のモデルに基づく障害回復機能を、LANで結合されたUNIXワークステーションに実装した。ただし一対のAPとLMは1つのワークステーション上の2つのプロセスとして実現している。具体的な実装法は次の通りである。

チェックポインティングの開始

各プロセスはほぼ正確な時計を持っているとする（これは同期したクロックを持っているということではない）。この時計から、定期的に各LMにほぼ同時に割込みがかかり、APの局所状態及びAPにつながるチャネル上のメッセージの記録が開始される。APは局所状態の記録が終わるまでブロックされる。

チャネルに対する送受信

ほぼ図2の通りに行う。マーカを受け取ったときは、LMはAPにメッセージをマーカの指示する長さだけバッファに読み込ませ、それを記録する。（こうすることにより、APが要求するメッセージの長さに関係なく、マーカが到来してから最初の受信手続きでチャネルの記録を完了することができる。）また、マーカが、その後にAPが送ったメッセージよりも先にLMに届くのを保証するため、マーカを受け取ったLMは確認を返す。マーカを送ったLMが確認応答を受け取るまで、APのそのチャネルへの送信はブロックされる（その他の処理はブロックされない）。

障害の検出とロールバック・リカバリー

チェックポイントにおいてマーカを送ってこないプロセスを、障害が発生したものと判定する。障害を検出したプロセスは、すべてのプロセスに回復動作のための準備をするよう通知し、故障したプロセスの回復を待つ。回復したプロセスは、自分がどのチェックポイントにおける記録を用いたかをすべてのプロセスに伝え、他のプロセスもそれを用いて回復する。

5 性能評価

4で述べた方法により障害回復機能を実装し、簡単なアプリケーション（2つのプロセスがただメッセージをやりとりするというもの）に対してその動作を確認した。障害の発生しない場合の性能を評価するため、このアプリケーションを回復機能のための変更なしで実行し、回復機能付きの場合と実行時間を比較した。その結果を図3に示す。測定したときのパラメータはチェックポイントの時間間隔であるが、一回のチェックポイントから次のチェックポイントまでに行われたチャネルに対する送受信回数との関係として示した。測定した最大の時間間隔は100秒であり、このとき実行時間は8%増加した。単位時間当りの送受信回数が一定のアプリケーションでは、結果は双曲線に近付くと予想されるが、図3の結果はそれに符合している。

ここで示した数値をそのまま一般化することはできないが、この実験用のアプリケーションは単位時間あたりの通信回数が最も多い場合の1つで、各プロセスの独立性が高い場合は、通常時の性能低下はもっと小さくなると考えられる。

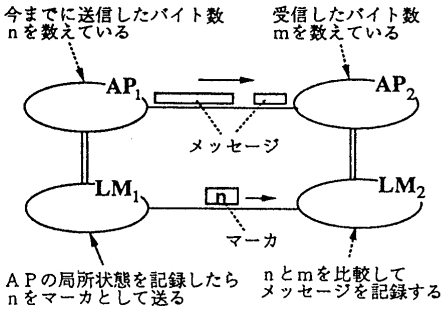

図 2: チェックポイントにおけるメッセージの記録

6 おわりに

以上、まず分散システムにおいて障害回復のためのアルゴリズムを実装する際に、解決すべき課題について考察し、図1のようなモニタシステムの一機能として実装しようとする場合の利点について説明した。また、実際に障害回復機能を実装し、特に通常時においてアプリケーションが受ける影響について、シミュレーションによる評価を行い、その結果を示した。今後の方針としては、実装したプログラムの改良、より多方面からみた性能評価、そしてより効率的なロールバック・リカバリーの実装、などを行いたいと考えている。

参考文献

[1] R. Koo, and S. Toueg, *Checkpointing and Rollback-Recovery for Distributed Systems*, IEEE Trans. on SE, Vol.SE-13,No.1,pp.23–31 (1987).

[2] Z. Tong, R. Y. Kain, and W. T. Tsai, *Rollback Recovery in Distributed Systems Using Loosely Synchronized Clocks*, IEEE Trans. on Distributed and Parallel Systems, Vol.3,No.2,pp.246–251 (1992).

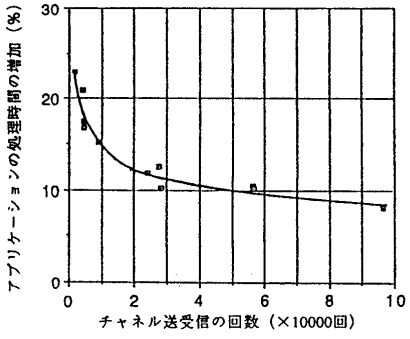

図 3: チェックポイントの間隔とアプリケーションの実行時間の関係