5F−6

モニタ用プロセッサを用いた分散システムのモニタリング手法に関する検討

増岡 義政, 相田 仁, 齊藤 忠夫
東京大学 工学部

1 はじめに

疎結合マルチプロセッサシステム（以下分散システム）では、処理の負荷あるいは機能が複数のプロセッサに分散されることにより、処理速度や信頼性の向上が図られる。しかし、統一のクロックを持たず、ネットワークを介した通信により協調動作を行うという構造のため、システム全体としての状態（グローバル状態）を簡単に判定できず、アプリケーションがユーザの期待通り動作するよう制御するのが難しいと考えられている。

分散システムにおいてグローバル状態を判定する方法としては、スナップショット・アルゴリズムのような一般性の高いものから特定のグローバル状態だけを検出するためのものまで、多数のアルゴリズムが存在するが、これらの手法をどのように実装し、またどのようにして実際の制御に役立てるかといった点については十分に明らかではない。そこで本稿では、分散アルゴリズムを実装して、多様な分散アプリケーションの制御を行うことのできるモニタシステムに要求される仕様について考察し、それをふまえて考案した分散モニタシステムについて、その概要を紹介する。またこのモニタシステムをLANで結合されたUNIXワークステーション上に仮想的に実装して、その性能評価を行った結果について示す。

2 分散システムのモニタリング

デッドロックの検出など、分散モニタシステムが提供する機能（以下モニタ機能と呼ぶ）には、一般に次のような特徴がみられる。

実時間性が高い

分散アプリケーションのモニタシステムにおいては通常複数のモニタ機能が共存しなくてはならないから、実装するモニタ機能の実時間性を考慮して各機能のスケジューリングを行う必要がある。

モニタ機能はアプリケーションにとって必須のものではない

従って、モニタシステムには、アプリケーションの動作を妨害しない（侵襲度（intrusion）をなるべく小さくする）ような構造が要求される。

アプリケーションによってモニタシステムに求められる処理の内容が異なる

アプリケーションによっては、モニタ機能のうち必要でないもの、またより効率的に実装できるものが存在する。従って、汎用のモニタシステムでも、分散アプリケーションの開発者が、いくつかのモニタ機能を取り外したり、独自にプログラムした機能を装着したりといったカスタマイズが、ある程度容易にできることが望ましい。

実際にモニタシステムを構築する場合には、以上の点に留意する必要がある。

3 モニタ用プロセッサを用いたモニタシステム

2で述べた要求を満たすモニタシステムとして、モニタ専用のプロセッサと共有メモリを用いるものを考案した。その基本的な構成は次の通りである。なお以下では簡単のため、1つのプロセッサには分散アプリケーションの1つのプロセスが走行しているとする。

- アプリケーションの各プロセス（Application Process(AP)）が走行している1つ1つのプロセッサに対し、モニタリングのための専用のプロセッサを図1のように密結合させ、その上にAPをモニタするプロセス（Local Monitor(LM)）を走らせる。
- 各LMは正確な時計を持ち、これをタイマとして用いてモニタ機能をほぼ同時に切り替えて実行していく（図2）。すなわち、APの挙動を常時監視するのではなく、定期的に検査する形となる。

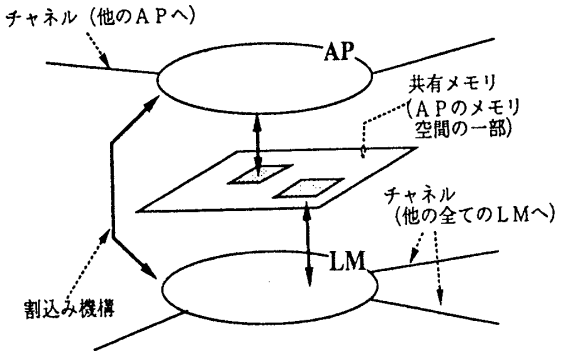

図 1: 分散システムモニタのハードウェアモデル

"A Study on Monitoring Distributed Computer Systems Using Monitoring Processors"
Yoshimasa Masuoka, Hitoshi Aida, and Tadao Saitoh
The University of Tokyo

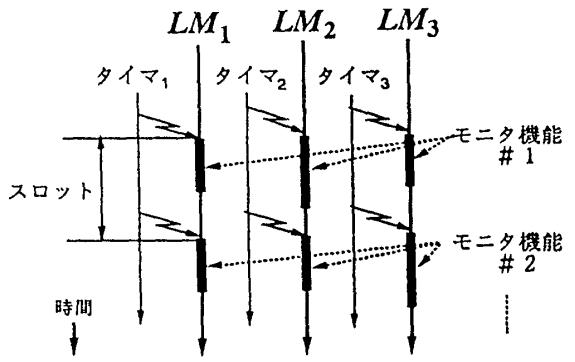

図 2: LM における各モニタ機能の起動

モニタ専用のプロセッサを用いるモニタシステムのモデルとしては、AP の状態に関する情報を、システムバスを常時監視することによりハードウェア的に集める方式も提案されている [1] が、アプリケーションに対する侵襲度をほとんど零にできる反面、分散アプリケーションに応じたカスタマイズが困難になってしまうと考えられる。その点を考慮して共有メモリを用いることを考えた。また、1つ1つのモニタ機能は図2のように自分に与えられたタイムスロットの間プロセッサを占有するので、実時間性の高い機能を実装することができ、さらに機能単位で LM の動作をカスタマイズしやすいという利点がある。一方、この方式では、各機能は自分のタイムスロットの間に処理を完了しなくてはならず、また LM のプロセッサ資源の無駄が多いという欠点がある。

4 UNIX ワークステーションへの実装

前節で述べた基本的構成を持つ分散モニタシステムのプロトタイプを、Ethernet 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いる。現在までにこのシステムに実装したモニタ機能は、

- 分散アプリケーションのチェックポインティング [2]
- デッドロックの検出

また、スロットに搭載されるのではなく、障害の発生などに応じて非同期的に起動される機能として、

- 分散アプリケーションの（最新のチェックポイントへの）ロールバック・リカバリー
- 全 AP への一斉割込み機能を用いた、アプリケーションの制御

を実装した。これらの機能を非同期的に起動した場合はタイマは止められ、処理の終了後再び合わせ直される。

以上の機能は一般の分散アプリケーション（ただし現状ではチャネル構成を走行中に動的に変化させるようなプログラムには使用できない）に対して、そのソースプログラムに最小の変更を加えるだけでアプリケーションのモニタリングが行なえるようになっている。LM においてスロットの構成は関数へのアドレスの配列として記述されており、この配列の内容を変えることで LM の動作をカスタマイズすることができる。

5 性能評価

アプリケーションに対する侵襲度（モニタを装着することでアプリケーションが受ける影響の度合。ここでは応答時間の増加率で表すことにする）は、モニタシステムの性能を議論する上で重要な要素である。そこで、簡単なアプリケーション（2つあるいは3つの AP がメッセージを一定回数往復させるというもの）について、モニタシステムを付加した場合の応答時間を測定し、付加しない場合と比較した。その結果を図3に示す。モニタを付加した場合の応答時間は、AP と LM が並列に動作する場合（LM が行なう処理のうち、AP と並列に行なえるものを省くことにより仮想的につくり出した）と、1つのプロセッサ上で動作する場合の2つについて測定した。主にチェックポインティングの際のディスクへの書き出しが並列化できるため、マルチプロセッサで実現すれば実用的なスロット長でも侵襲度は十分小さいといえる。

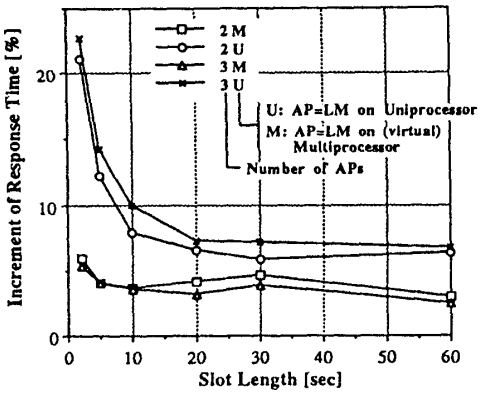

図 3: スロット長と侵襲度（応答時間の増加）の関係

6 おわりに

以上、分散アプリケーションのモニタリングを行なう際に要求される仕様について考察し、それに基づいて考案したモニタシステムの基本的構成とその実装法を紹介した。さらに簡単なアプリケーションに対して応答時間を比較した結果を示した。今後も専用プロセッサと共有メモリを用いてモニタリングを行なうことの有用性を確かめるために、引続き性能評価およびモニタ機能の追加を行ない、さらにより一般のアプリケーションのモニタリングが可能となるようにシステムを改良していきたいと考えている。

参考文献

[1] D. C. Marinescu, J. E. Lumpp,Jr., T. L. Casavant, and H. J. Siegel, *Models for Monitoring and Debugging Tools for Parallel and Distributed Software*, J. of Parallel and Distributed Computing, Vol.9, pp.171-194 (1990)

[2] 増岡, 相田, 齊藤, "分散システムにおける障害回復法に関する検討", 情報処理学会第 45 回全国大会講演論文集 2P-09