3D－4

機能分散マルチプロセッサリアルタイムシステムにおけるマイグレート可能なグローバルタスクの導入

鍋田　努　　石川　知雄
武蔵工業大学情報通信研究室

高田　広章
豊橋技術科学大学情報工学系

1. はじめに

近年，交換機やネットワークルータなどの通信制御システム等の分野を中心に，高性能なリアルタイムシステムに対する要望が高まっている．これらの応用では，システムに接続する入出力装置の増設や，システムに対する要求性能を上げるといった仕様変更により，プロセッサの数を増やす必要のある状況が多く，その際にシステムの設計変更を最小限に止めたいという，スケーラビリティに対する要求がある．現在，リアルタイムシステムのためのスケーラビリティに重点を置いた研究として，マルチプロセッサリアルタイムOSであるITRON-MPを用いて様々な研究がなされているが[2][3]，それらは現状ではプロセッサ間を移動するタスク，すなわちタスクマイグレーションに関しては実装していない．

本研究では，このITRON-MPを題材にし，スケーラビリティの要求に答えつつ，プロセッサ間をマイグレートするタスクを実装することを目的とする．時間要求の厳しくないタスクをアイドリング状態のプロセッサ上で実行することにより，時間要求の厳しいタスクの応答時間を悪化することなく，システム全体のスループットを向上することが本研究の目的である．

2. 研究背景

2.1. マルチプロセッサと共有資源

マルチプロセッサ環境において，タスクはメモリ等の共有資源を持ち，共有資源のアクセスについて，スピンロックなどの手段による排他制御が必要である．排他制御によりデータの一貫性は保証されるが，他のタスクがアクセスしている資源を操作するには，その資源が解放されるのを待たなければならない．したがって，他プロセッサ上のタスクと同期・通信を行うタスクの処理については，競合するプロセッサ数が増えるにつれて最大実行時間が長くなるのは本質的に避けられない．時間要求の厳しいリアルタイムシステムを扱う場合，最大実行時間の延長は大きな問題である．スケーラビリティの向上を図るためには，この問題を解決しなければならない．

2.2. タスクの分類

前述の問題を，タスクを分類することによりプロセッサ数に対する依存を極力おさえる，という研究がなされてきた[1]．その研究において，タスクは次のように分類されている．

- プライベートタスク

操作できるカーネル資源(タスクやセマフォなど)に制限をもたせ，他プロセッサと同期・通信を行わないタスク．自プロセッサ内のカーネル資源については優先的に操作できる．自プロセッサに閉じた処理をプライベートタスクで実行することで，そのタスクの最大実行時間はプロセッサ数に依存せずに決まる．

- ローカルタスク

他プロセッサと同期・通信を行うタスク．他プロセッサのプライベートタスクとは通信できない．最大実行時間はプロセッサ数のオーダーで長くなる．

- グローバルタスク

プロセッサ間をマイグレートできるタスク．他のグローバルタスクや他プロセッサのローカルタスクとは通信できるが，プライベートタスクと通信することはできない．

タスクの優先度は，高い順に，プライベートタスク，ローカルタスク，グローバルタスクである．また，タスクだけでなく，セマフォなどのカーネル資源も同様に分類されている．

現在，グローバルタスクについては実装が行われていない．プライベートタスク，ローカルタスクは実行するプロセッサが固定されており，アイドルプロセッサを有効に活用できない．本研究ではグローバルタスクを実装し，リアルタイムOSにおいて，スケーラビリティの要求に答えつつ，タスクのマイグレーションを導入する．

Introduction of migratable Global task on function distributed multiprocessor real-time system
Tsutomu Nabeta, Tomo Ishikawa
Information and Communication Laboratory, Musashi Institute of Technology
Hiroaki Takada
Department of Information and Computer Sciences, Toyohashi University of Technology

3. 実装方法

3.1. グローバルタスクの管理方法

ITRON-MP では実行可能状態のタスクを管理するためにレディキューが用意されており，タスクの切り替えなどに利用されている．

ローカルタスクやプライベートタスクと異なり，グローバルタスクは同時に複数実行されることが考えられる．実行中のタスクをレディキューにつないだままにすると，タスクの切り替えを行う際，次に実行するタスクをキューの先頭から探していかなければならない．プロセッサ数が増えて，グローバルタスクが同時に多数実行されている状態では，このオーバーヘッドは大きくなることが考えられる．

3.2. ランキューの導入

そこで，実行中のタスクを管理するランキューを導入する．高い優先度のタスクが到着したとき，最も優先度の低いタスクを実行しているプロセッサを横取りしなければならないので，ランキューは優先度の低い順に実行中のタスクをつなぐものとする．

新たなタスクが実行可能状態になったとき，それがグローバルタスクだったときは，ランキューの先頭のタスクからプロセッサを横取りすれば良い．ローカル/プライベートタスクだった場合，ランキューの中からそのタスクを実行するプロセッサを探し，タスクを切り替えなければならない．例えばランキューが図 1 のような状態で(数値が小さいほうが高優先度)，プロセッサ 2 でプライベートタスク T4 が実行可能状態になったとする．このとき，

1. T1 を止めて P1 に T2 をマイグレートし，T2, T3, T4 が動いた状態にする．
2. T2 を止めて，T1, T3, T4 が動いた状態にする．

という，2 つの選択肢がある．

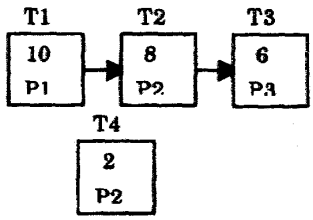

図 1 ランキューと，実行可能状態になったタスク

1 の実装は優先度的には理想であるが，処理の短いタスクが複数のプロセッサで交互に起床した場合，タスクが頻繁にマイグレーションをおこし，性能が低下する恐れがある．また，ランキューは共有資源なので，ランキューに書き込みを行う場合はロックを獲得しなければならないが，それでは処理時間が競合するプロセッサ数に依存してしまい，プライベートタスクの特徴が損なわれてしまう．よって，プライベートタスクについては 1 の実装は適切でない．2 の実装では一時的に優先度逆転がおこり，それがいつ解消されるかが問題になる．

3.3. 優先度逆転の解消

1 つは，最も優先度の低いグローバルタスクを実行しているプロセッサに定期的に優先度逆転をチェックさせる方法が考えられる．この場合，チェックにかかるオーバーヘッドにより，性能が低下する恐れがある．もう 1 つは，優先度逆転をチェックするシステムコールを用意して，ユーザが必要に応じてそれを実行することにより，優先度逆転を解消する方法が考えられる．

いずれの方法でも，横取りが発生したときに，(図 1 の例では T2 に)横取り発生のマークをする必要がある．優先度逆転のチェックをするプロセッサは，このマークを見て必要に応じ，タスクを切り替えれば良い．このマークを書き込む処理は，現在実行中のタスクに対してそれを実行しているプロセッサだけが行う処理なので，ロックを獲得する必要はない．よって，プライベートタスクの最大実行時間がプロセッサ数に依存しないという特徴は保たれる．

4. むすび

現在 ITRON-MP に Global タスクを実装する作業を行っている．タスクのマイグレーションが頻繁に発生する問題や，優先度逆転の時間等を，実機を用いた性能評価により検証し，よりよい実装方法を検討する．また，アイドルプロセッサに関し，それをランキューにつないでおく方法や，プロセッサの数だけアイドルタスクを用意して，それをグローバルタスクに分類することによりタスク切り替えを簡略化する方法も検討中である．

参考文献

[1] 高田広章，坂村健，"非対称マルチプロセッサシステムのためのスケーラブルなリアルタイムカーネルの構想"，信学技報(1995 年実時間処理に関するワークショップ RTP'95)，vol.94，no.573，pp.1-8，電子情報通信学会，Mar. 1995.

[2] 高田広章，坂村健，"中断可能なキューイングスピンロックアルゴリズム"，電子情報通信学会論文誌(D-I)，vol.J78-D-I,pp.661-669,Aug.1995

[3]王才棟，高田広章，坂村健，"優先度継承スピンロックアルゴリズムの性能評価"，信学技報(1996 年実時間処理に関するワークショップ RTP'96)，電子情報通信学会，Mar.1996.