4K−11

軽量プロセスと非完了システムコール機能の比較
— 並列処理の性能評価 —

箱守 聰, 横山 和俊, 谷口 秀夫
NTTデータ通信（株）開発本部

1 はじめに

プロセッサの負荷が大きい処理を効率的に実行するためには、同時に実行可能な部分を並列化することが重要である。処理の並列化には、プロセスを複数生成する方式と、非完了システムコール機能を用いる方式とがある。前者の場合、より効率的な処理のためには、プロセス間でその構成要素を互いに共有する、プロセスの軽量化が必要となる。

筆者らは、プロセスの軽量化方式について検討しており[1]、そのうちのいくつかの方式について、分散型リアルタイムオペレーティングシステム(DIROS: DIstributed Real Time Operating System)上に実現した。本稿では、非完了システムコール機能を利用したプロセスと、軽量プロセスとを用いて並列処理を実現した場合の比較結果について報告する。

2 並列処理への適用性

プロセスの軽量化については、プロセス間でどの要素を共有するかによって様々な形態が考えられる。ここで対象とした軽量プロセスは、テキスト部, データ部, 資源アクセス管理テーブルおよびメモリ管理テーブルをプロセス間で共有したものである。

非完了システムコール機能は、1つの処理要求を、処理受付コールと結果取得コールとに分けて提供する機能である[2]。このため、1つのプロセスが同時に複数の処理要求をOSに発行することができる。

並列処理を、軽量プロセスや非完了システムコール機能を利用したプロセスで実現した場合、従来のプロセスを用いた場合に比べ、以下の長所がある。

(1) 並列に動作する処理の間でデータ部を共有するため、データ転送を高速に行なうことができる。従来プロセスを用いる場合は、プロセス間でデータの転送を行なう際にデータコピーが発生する。

(2) 軽量プロセスは1つの論理空間に共存し、非完了システムコール機能は1つのプロセスで並列処理を実現するため、どちらも並列処理中に論理空間の切り換えが発生しない。従来プロセスを用いる場合は、プロセス切り換え時に論理空間の切り換えが発生し、メモリキャッシュやTLBのヒット率が低下する。

(3) 並列に動作する処理の間で資源アクセス管理テーブルを共有するため、同じ資源に対するアクセス情報を共有できる。これは例えば、1つのファイル内のデータ検索を並列化しようとする場合に有効である。従来プロセスを用いると、アクセス情報を共有するために同期をとる必要がある。

また、以下の短所がある。

(4) 並列に動作する処理の間で、メモリ管理テーブルと資源アクセス管理テーブルを共有しているため、これらの間でデータの排他制御が困難である。また、処理の単位毎で固有に持つ領域への不当アクセスを防ぐことができないため、実行時の信頼性が低下する。

軽量プロセスを用いる並列処理と、非完了システムコール機能による並列処理とを比較する。前者では、並列処理の単位間で同期をとるために軽量プロセス間で通信が必要になる。後者では、同一プロセス内の処理となるため、通信は必要ない。

3 性能評価

3.1 処理例

並列処理として、「処理Aを実行した後、その結果を用いて処理Bを並列化して行なう場合」について考える。軽量プロセスと非完了システムコール機能による処理例を図1に示す。

図1 (a)は、非完了システムコール機能を用いて並列処理を行なう場合の、処理の流れである。このとき、処理は全て1つのプロセス上で行なわれる。プロセスは、並列度の分だけ処理要求システムコールを発行すると、結果取得のシステムコールを発行して、それらの処理の終了を待ち合わせる。

軽量プロセスを用いて並列処理を行なう場合の、処理の流れを図1(b)に示す。処理Aを実行する親軽量プロセスと、複数の処理Bを実行する並列化軽量プロセスによって処理が行なわれる。これらのプロセスは、メッセージ通信により同期をとる。

3.2 処理速度

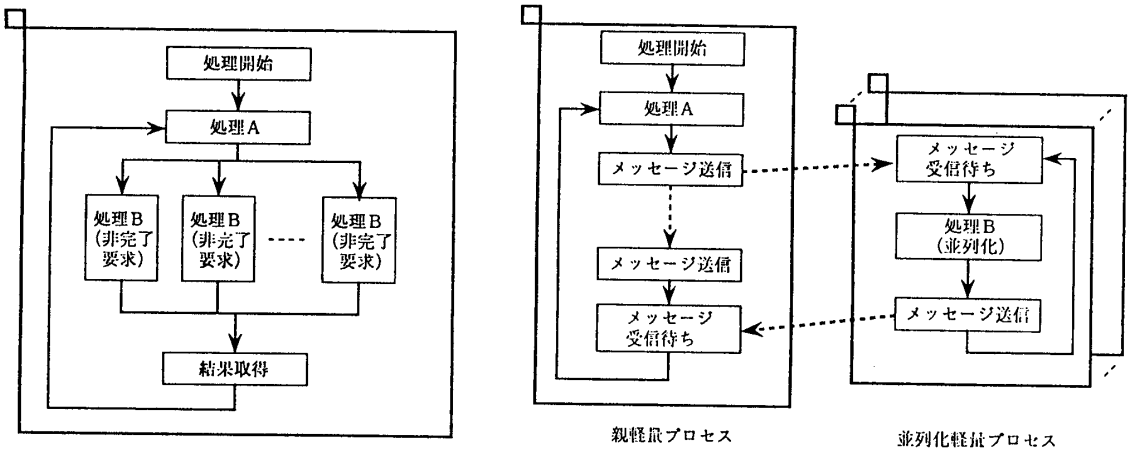
軽量プロセスと非完了システムコール機能を用いた場合について、並列処理の実行時間の違いを評価した。

評価プログラムは、3.1節の処理例をモデルとした。ここで、処理Aはfor文によるループとし、処理時間はT_Aである。処理Bについては、軽量プロセスの場合は、一定時間プロセスをwaitにするシステムコールとし、非完了システムコール機能の場合は、一定時間後に結果を返却する非完了システムコールとする。この時間は

Parallel Processing Performance with Light Weight Process and Asynchronous System-call Function

Satoshi HAKOMORI, Kazutoshi YOKOYAMA
and Hideo TANIGUCHI

NTT DATA COMMUNICATIONS SYSTEMS CORPORATION



(a) 非完了システムコール機能を利用したプロセスの場合

(b) 軽量プロセスの場合

図1．並列処理の例

T_Bである。また、軽量プロセスの場合のプロセス間通信については、並列化軽量プロセスから親軽量プロセスへは、どのプロセスからの通信か識別する必要があるため、メッセージ通信を用いる。親軽量プロセスから並列化軽量プロセスへの通信は、同期のみが必要なため、セマフォを用いる。

これらの処理を一定回数繰り返し、処理Aを開始してから、処理Bが全て終了するまでの時間を測定した。$T_A : T_B$を約1：9とし、並列処理の数nを変化させた時の、処理時間の変化を図2に示す。

これより、以下のことがわかる。

(1) $n = 1$のときの処理時間の内訳は、(Aの処理時間)+(Bの処理時間) +(同期・切り換えに要する処理時間)である。軽量プロセスを用いる場合の方が処理時間がかかるが、これはプロセス間通信によるオーバーヘッドである。

(2) $n = 2 \ldots 8$程度までの間は、処理速度は$n = 1$の時の約$1/n$となる。ここで、軽量プロセスを用いる場合の方が減少が緩やかであるが、この原因には、並列処理の依頼から実行開始までの時間の差が考えられる。軽量プロセスを用いる場合は、他の軽量プロセスに対して処理依頼を通知しても、現在走行中の軽量プロセスが切り換わるまで、通知を受けた軽量プロセスは処理を開始できない。非完了システムコール機能を用いる場合は、処理要求システムコールを発行した時点で、処理が開始される。したがって、処理の要求から実際にその処理の開始までは、軽量プロセスを用いる方が時間がかかり、これが全体の処理時間に影響する。

(3) $n > 8$程度になると、並列処理の数にかかわらず処理時間は一定となる。これは、$T_A : T_B = 1 : 9$であることから、$n \sim 9$以上では処理Aが常に動いている状態となり、処理Bを終了した軽量プロセスや非完了システムコールの結果通知が、処理Aの終了を待つことになるからである。したがって、この処理ではこれ以上の並列化は有効でない。

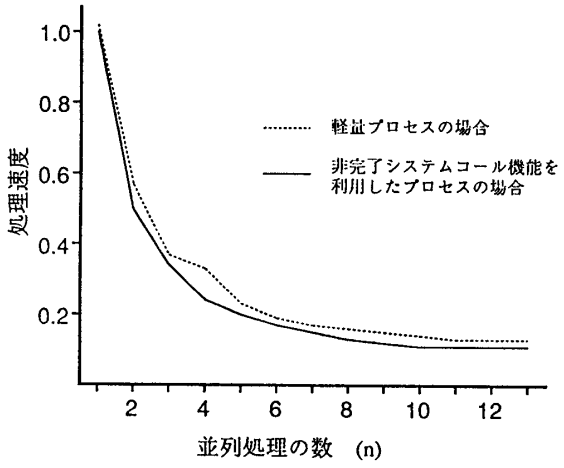


図2．処理時間
(処理速度は、並列処理の数が1の時の非完了システムコール機能による場合を1とする)

4 おわりに

並列処理の効率化を行なう際に、軽量プロセスと非完了システムコール機能を用いる場合の処理の特徴を比較し、処理時間の測定を行なった。

軽量プロセスを用いる場合は、非完了システムコール機能を用いる場合に比べて、プロセス間通信によるオーバーヘッドや、処理の依頼と処理を開始する時間の差により、並列処理が遅くなる。

参考文献

[1] 谷口, 横山, 箱守: “プロセスの軽量化に関する検討”第42回情処全大(1991).

[2] 谷口, 遠城, 井村, 境: “分散型リアルタイムオペレーティングシステム:DIROS,” 情処研報, 89-OS-45-9(1989).