汎用超並列 OS *SSS–CORE* におけるスケジューリング方式の評価

信国 陽二郎　　松本 尚　　平木 敬

東京大学大学院理学系研究科情報科学専攻

1　はじめに

分散メモリ環境では、メモリへのアクセスコストが距離によって異り、並列プロセスの効率的実行の実現には相対的にコストの高いアクセスを減らすことが求められる。複数のプロセスが動作する汎用的環境ではその実現方法として、メモリページなどの実資源の使用状況を考慮したスケジューリングを行ないシステム全体の性能を上げることが可能である。またアクセスコストの小さなメモリページから置換を行ない、再アクセス時のコストを抑えることも、全体の性能向上に寄与する。

本稿ではメモリアクセスベースの確率モデル上で、具体的なメモリ管理方式 / アクセス頻度 / アクセスコストを付加したシミュレーションにより、並列プロセス毎に所有する実ページ情報を利用したスケジューリング法、及びメモリ置換方式の評価を行う。

2　SSS–CORE

SSS–CORE [1] は NUMA 型並列分散計算機を対象とした汎用並列 OS である。時分割 とパーティショニングを併用してマルチユーザ / マルチジョブ環境を提供し、並列アプリケーションの効率良い実行を実現することを目的とする。

SSS–CORE では、計算機の相互結合網の階層構造と一致した構造の資源管理構造を用意して資源情報の管理をする。2 レベルスケジューリングを採用し、カーネルはこの資源情報と各プロセスから提示されるスケジューリング制約にしたがって、プロセス単位に資源の割当を行なう。

3　シミュレーション・モデル

3.1　主な特徴

これまでの研究で利用したモデル [2] を拡張し、OS レベルのシミュレーションのため可能な限り簡略化されたモデル [3] を構築した。対象とするのは分散メモリ環境である。メモリとプロセッサの対をクラスタとし、相互結合網は木構造である。通信等のコストは通過するネットワークの段数に比例し、基本コストをパラメータとして与える。ネットワーク上の競合は無視する。

各プロセスには共有空間と非共有空間を、それぞれ別々のページ単位のメモリ参照頻度表により与える。一クラスタにおける並列プロセスの実行コンテキストをスレッドと呼び、各スレッドは異なった空間を与えられる。スケジューリングにより割り当て位置が変化した場合には、ローカルページはネットワークを介して on-demand に移動する。各プロセスは要求プロセッサ台数と同数のスレッドにより構成される。並列度は一定で、生成時から終了時まで変化しない。ページがリモートにも存在しない場合には、ディスクアクセスを生じる。共有空間に対しては分散共有メモリシステムを構築し、プロセスの各スレッドは参照頻度表を共有する。コヒーレンシ管理はアップデート方式で、ページは置換されない限りメモリに残る。また、メモリ参照コストはクラスタ内 $\ll$ クラスタ間 $\ll$ 2 次記憶であると仮定する。

プロセスの実行はクロックベースの確率モデルで、可能であれば毎クロックリード又はライトのメモリ参照を行なう。またプロセスのスレッドは、パラメータで与えられた時間間隔でランダムな組合せで同期を起こす。ここで有効実行 (時間 / クロック) とは、ページ待ち / 同期待ち以外の動作 (をした時間 / クロック数) のことである。表 1に今回のシミュレーション条件を示す。

Table 1: 各種コスト及びパラメータ

項目	値
ディスクアクセス	10000 clk
リモートアクセス	100 clk
通信	10 clk
プロセッサ数	16 台
1 メモリのページ数	400 pages
1 ページのサイズ	4096 Byte
総メモリ量	25.6 Mbyte
1 quantum	100000 clk

3.2　スケジューリング法

シミュレータ上に 5 つのスケジューリング法を実装した。タイムスライス毎に資源の占有 / 消費状況に応じた優先度計算 [2] を行ない、優先度の高いプロセスから順に、それぞれ以下に示す方法で資源管理木構造を利用してスケジューリングを行なう。以下でホームノードとは管理木構造上の、プロセスの要求台数以上のプロセッサを含む最下位レベルのノードで、前回の割り当て領域を代表するものである。

algo4 ページのあるクラスタのみを割り当てる。

algo3 ホームノード以下の領域でページのあるところから割り当てる。

algo2 初めにホームノード以下の領域に割当を試み、失敗した場合にはルートノード以下の領域で、自分のページが存在する領域から割り当てる。

algo1 資源管理木上の端から順に連続したプロセッサを必要なだけ割り当てる。

algo0 ランダムに必要な数だけプロセッサを選択し割り当てる。

プロセスに割り当てられたクラスタへのスレッドのスケジューリングは、以下の通りである。

An Evaluation of Scheduling Methods for General Purpose Operating System *SSS–CORE*

Yojiro NOBUKUNI, Takashi MATSUMOTO, Kei HIRAKI

nobukuni@is.s.u-tokyo.ac.jp

Table 2: プロセス・セット

#DP	#PS	非共有空間サイズ * プロセス数
4	4	$30pages*2, 80pages*2$
8	4	$30pages*4, 80pages*1$
16	2	$30pages*2$
総並列度 80		総非共有空間 3200pages

#DP: 要求プロセッサ数, #PS: プロセス数

Table 3: 1 段 16 進構成で RA を増加させた結果

RA	置換方式	algo0	algo1	algo2	algo3	algo4
100	方式 1	46.20	49.12	46.07	46.07	58.64
	方式 3	85.93	87.99	90.41	90.41	95.15
300	方式 1	39.18	36.28	42.01	42.01	56.92
	方式 3	64.69	71.75	78.17	78.17	90.21

RA：リモートアクセス・コスト、値は有効実行率 [%]

- 前回の割当と重なるクラスタには同じスレッドをスケジュールする。
- それ以外のクラスタには、残りのスレッドを順にスケジュールする。

3.3 ページ置換方式

ページの置換は、再び参照される可能性の低いもの、また再アクセスのコストが小さいものから置換の対象とした方が性能向上に寄与する。したがって、一般に相対的にアクセス頻度の高いローカルページよりも共有ページ、またコピーを持たない共有ページ (以降便宜的にラストワン・ページと呼ぶ) よりも共有コピーページから置換の対象とする。また現在実行中のプロセスよりはその他のプロセスに属するページの方が参照される可能性が低い。そこでページの種類により、次のように置換対象とする順序を与えることができる。

1. 他のプロセスの共有コピーページ
2. 実行中プロセスの共有コピーページ
3. 他のプロセスのラストワン・ページ
4. 他のプロセスのローカルページ
5. 実行中プロセスのラストワン・ページ
6. 実行中プロセスのローカルページ

今回は、次の二つの置換方式の比較を行なった。

方式 **3** 優先度の低いプロセスから先に、上記のページの種類順に選択。

方式 **1** ページの種類 / プロセスの区別なしに、単純な LRU 順。

ただし両者とも、一貫性処理中の共有ページの場合は置換対象としない。

4 結果及び考察

前述のモデルを用いて表 1のようなパラメータによるシミュレーションを表 2にあげたプロセスの組合せに対して行なった。各プロセスはどれも 80 ページの共有空間を持ち、 1000 有効実行クロック毎に同期する。図 1がその実行結果である。各スケジューリング法に対して、各ネットワーク形状において両メモリ置換方式を採用した時の、システム全体の有効実行率を縦軸に示した。なおデータの取得はシミュレーションを開始後、一つ目のプロセスが終了 (20 タイムスライス分の有効実行) した直後のタイムスライスに行なった。

図 1のグラフは概ね右上がりで、スケジューリング方式 algo4 が最良の結果を示している。またメモリの置換方式についてみると、ネットワークの形状とスケジューリング方式のどの組合せの場合にも、置換方式 3 が方式 1 に勝っている。

表 3はネットワークが 1 段の場合に、リモートアクセスのコストを 300 とした実験結果である。ネットワーク上のコストが増加すると、スケジューリング方式間の性能差が広がる。ネットワーク上の競合を無視しているため、競合の起きやすいこのようなフラットな形状の場合には、リモートアクセスのコストが 100 の場合の性能差は、実際よりも小さくなっていると推定される。

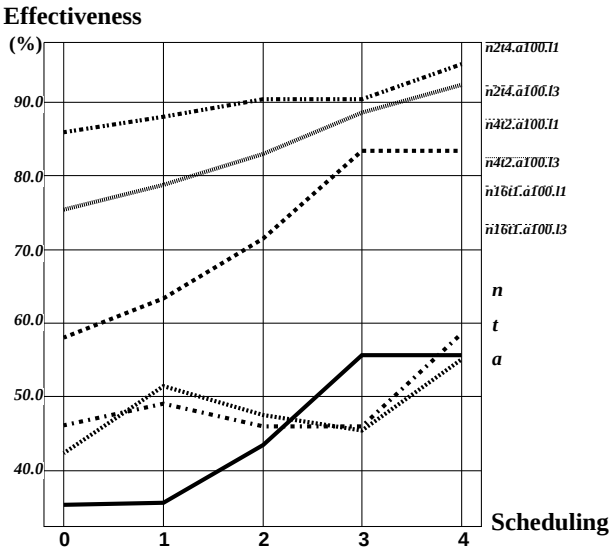

Figure 1: シミュレーション結果

5 おわりに

シミュレーションにより、プロセスのメモリ参照時のコストをなるべく抑えるように、プロセスにはページを持つクラスタを割り当て、ページの移動や置換による悪影響を抑えると効率的な実行ができることが示された。以上の結果をもとに、 *SSS–CORE* の実装に取り掛かる。

謝辞

本研究は情報処理振興事業会（ＩＰＡ）が実施している独創的情報技術育成事業の一環として行なった．

References

[1] 松本尚, 古荘進一, 平木敬. 汎用超並列オペレーティングシステム SSS–CORE. 日本ソフトウェア科学会第 11 回大会論文集, pp. 13–16, October 1994.

[2] 信国陽二郎, 松本尚, 平木敬. 汎用並列ＯＳのための資源情報を利用したスケジューリング方式の検討. 信学技報, Vol. 95, No. 210, pp. 111–118, August 1995.

[3] 信国陽二郎, 松本尚, 平木敬. 並列ＯＳの性能予測を可能にするシミュレーションモデル. 情処研報 96-ARC-117, Vol. 96, No. 23, pp. 19–24, March 1996.