アクティビティ方式並列実行機構の研究

6B－4 —2レベル・スケジューリングを用いた複数ユーザ化の試み—

中山 毅　小林 伸治†　中山 泰一‡　永松 礼夫　森下 巖
(東京大学工学部)

1 はじめに

共有メモリ型並列計算機において多数の細粒度タスクを効率良く並列実行する機構として筆者らは、アクティビティ方式を提案した[1][2]。この方式により並列プログラムをより高い効率で実行できることが、実機においても確認されている[3]。

本稿では、並列計算機の能力を無駄なく使うために複数のユーザから多数のコンテクストが投入される状況を想定し、複数ユーザに対応するようにアクティビティ方式の拡張を行なった。

また、拡張したアクティビティ方式スケジューラの性能を、シミュレータ上で応用プログラムを実行することによって評価した。複数ユーザ対応としたために、単一ユーザの場合の処理時間が増加することが予想されるが、シミュレーション結果では、最悪の場合でも単一ユーザ処理時間の増加は1割程度に収まることが確認された。

2 アクティビティ方式

並列実行機構(スケジューラ)では、多数の細粒度タスク（実行処理単位）の実行要求を処理するオーバヘッドは、できる限り小さいことが望ましい。

しかし、オーバヘッドが比較的少ないとされる軽量プロセスを用いても、並列実行要求ごとに軽量プロセスを生成する方法では、プロセッサ時間やスタック領域の消費などが無視し得ない量となる。

アクティビティ方式では、実際に並列実行できるタスクの数は高々実プロセッサ数であることに着目し、予め各プロセッサに軽量プロセスを1つずつ用意しておく。実行要求システムコール

```
make_child(手続き, 引数)
```

によって、手続きと引数の組（アクティビティ）が作成されキューに保持される。実際の処理は、各軽量プロセス(すなわち各プロセッサ)が、未実行のアクティビティをキューから取り出して処理することを繰り返すことで行なわれる。

この方式では、生成される軽量プロセス数は実プロセッサ数と同じに留まり、無駄な軽量プロセス生成に伴うプロセッサ時間やスタック領域の浪費といったコストを小さくすることができる。

An Activity Based Parallel Execution Mechanism for Multiple Application Programs using 2-level Scheduling by TAKESHI NAKAYAMA, SHINJI KOBAYASHI, YASUICHI NAKAYAMA, LEO NAGAMATSU and IWAO MORISHITA (University of Tokyo).

† 現在 (株)富士通研究所 ‡ 現在 電気通信大学

3 複数ユーザへの対応

アクティビティ方式を複数ユーザに対応するよう拡張する場合、アクティビティ・キューは1つのままにして異なるユーザ(コンテクスト)から生成されたアクティビティを混在させて管理する方法と、コンテクストごとにアクティビティ・キューを持たせる方法とが考えられる。

しかし前者では、どのコンテクストの処理をどのプロセッサ（すなわち各軽量プロセス）が実行するかを全く制御できなくなり、最悪の場合、各アクティビティの実行ごとにコンテクスト切替のコストがかかる可能性がある。

いっぽう後者ならば、各コンテクストへプロセッサ能力を配分する方法の問題を独立させて考えることができ、コンテクスト内でのスケジューリングは単一ユーザの場合とほぼ同じになる。また、アクティビティ・キューの長さの変化に応じて、コンテクスト切替の回数をより減らす工夫もできると期待される。

上述の理由により、本稿では、後者の2段階スケジューリング[4]を採用した。まず、グローバルスケジューラと呼ぶプロセッサ配分プログラムが、図1に示すように予めプロセッサ数だけ用意した軽量プロセスを各ユーザへと割り当てる。なお、グローバル・スケジューラの実装については、比較的プロセッサ数やユーザ数の少ないモデルに対応することを考え、各コンテクストの状態を集中管理することができるように、専用に1つのプロセッサを割り当ててある。

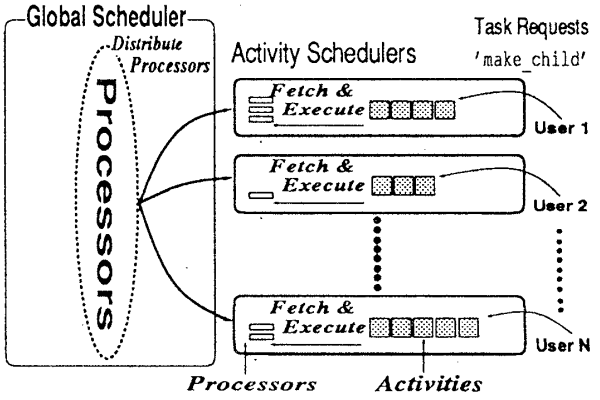

図 1: 複数ユーザ対応アクティビティ方式スケジューラ

4 実験

4.1 実験環境

拡張したアクティビティ・スケジューラの性能を評価するため、筆者らの研究室で開発したSPARCプロセッサのシミュレータを用いて実験した。バス結合型の理想共有メモリをシミュレートし、複数のコンテクストを持たせるためにメモリ管理機構（MMU）を持っている。

このシミュレータを用い、複数の6-queens問題を並行して解くプログラムを実行し、その処理時間とシステム処理やユーザ処理といった内訳を調べた。なお、プロセッサ数は4、8の2通りで実験を行なった。

4.2 グローバル・スケジューラの方針

グローバル・スケジューラのプロセッサ割り当て方針としては種々のものが考えられるが、ここでは、

- Everytime: 各プロセッサはアクティビティの実行を終了する毎に、タスクの存在する他のコンテクストのキューからアクティビティを取り実行する。
- Even: タスクの存在するコンテクストそれぞれに等しい数のプロセッサを割り当てる。
- Empty: 各プロセッサは現在実行中のコンテクストのアクティビティ・キューが空になるまでコンテクスト切替を行なわない。

といった方針のものを試作した。Everytimeはコンテクスト実行の公平性という意味ではもっともよい方針であるが、上記の中ではコンテクスト切替の回数が最大となり、そのコストも最大になる。

4.3 結果

図2にプロセッサ数8、Even方針の際の1ユーザあたり処理時間のグラフを示す。横軸はユーザ数で、比較のために、単一ユーザ対応スケジューラでの処理時間を左端に示してある。グラフ中、totalと示した線が、1ユーザあたりの総処理サイクル数、すなわち総処理時間を表しており、user、act_sch、contextは、それぞれ1ユーザあたりのユーザプログラム実行、アクティビティスケジューラ関連処理、コンテクスト切替処理に要したサイクル数を、idleは1ユーザあたりのプロセッサの累積アイドルサイクル数を表す。

このグラフより、拡張アクティビティ・スケジューラでは、ユーザ数が増えればプロセッサがより有効に利用される為アイドル時間が減少すること、その減少分がコンテクスト切替などによる処理時間の増加を上回ること、結果として1ユーザあたりの所要サイクル数が単一ユーザの場合よりも減少することが読みとれる。

また、図3に最もコンテクストの切替コストの高い、プロセッサ数4、Everytime方針の場合の1ユーザあたり処理時間のグラフを示す。今回の実験の中ではこの場合でのみコンテクスト切替コストの増加がアイドル時間の減少を上回り、ユーザ数2以上の処理時間が増加してしまった。しかし、ここには示さないが、他のスケジューリング方針の場合には、プロセッサ数が4であってもアイドル時間の減少分が、充分コンテクスト切替コストを補うという結果が得られている。

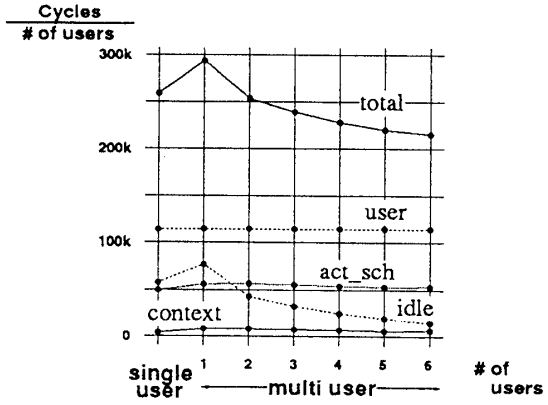

図 2: 1ユーザあたり処理時間（プロセッサ8、Even方針）

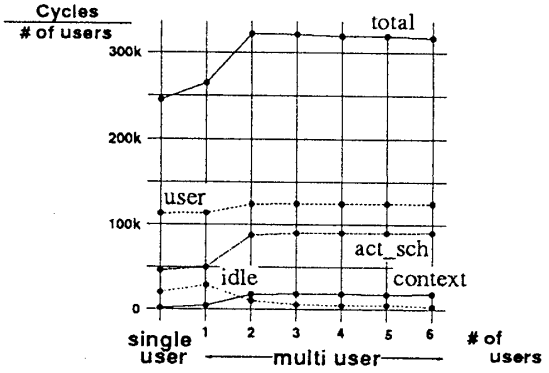

図 3: 1ユーザあたり処理時間（プロセッサ4、Everytime方針）

5 まとめ

複数のユーザに対応するように、アクティビティ方式の拡張を行ない、その性能を評価した。シミュレーション実験により、充分小さなコストで複数コンテクストへの拡張を実現できたことを確認した。今後、スケジューリング方針ごとの差異について、より詳細に検討していく予定である。

参考文献

[1] 田胡, 檜垣, 森下: 共有メモリ型並列計算機のためのアクティビティ方式を用いる並列実行環境, 情報処理学会論文誌, Vol.32, No.2, pp.229-236 (1991).

[2] 中山, 永松, 出口, 森下: 共有メモリ型並列機のための新しいアクティビティ方式並列実行機構, 情報処理学会論文誌, Vol.34, No.5, pp.985-993 (1993).

[3] 中畑, 本橋, 中山, 永松, 出口, 森下: アクティビティ方式並列実行機構の共有メモリ型並列機への実装と評価, 第46回情報処理学会全国大会論文集, 4F-1 (1993).

[4] 小林: 複数の並列応用プログラムをサポートするアクティビティ方式スケジューラの研究, 東京大学大学院工学系研究科情報工学専攻修士論文 (1993).