アクティビィ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

4F−1

中畑 昌也 本橋 健 中山 泰一 永松 礼夫 出口 光一郎 森下 巖
(東京大学工学部)

1 まえがき

共有メモリ型並列計算機において多数の細粒度のタスクを並列に実行する場合の効率の良い実行管理機構として、筆者らはアクティビティ方式[1]を提唱してきた。さらに基本の方式を改良した「遺言」方式[2]を提案し、高い効率で並列プログラムを実行できることをシミュレーションで確認した。

本稿では共有バス型の並列機上にこれらの実行管理機構を実装し、応用プログラムを走らせてその性能を評価した結果、実機においても「遺言」方式が有効であることを確認した。

2 アクティビティ方式

2.1 基本方式

タスクの生成要求が出されるたびに軽量プロセスを生成する実行機構においては、タスクが細粒度の場合のプロセス生成・消滅のコストが非常に大きい。それに対してアクティビティ方式では、あらかじめプロセッサ台数と同じだけ用意した軽量プロセスを繰り返し使用することにより軽量プロセスの生成・消滅のコストを削減する。具体的には、並列実行可能なタスクの発生の際、発生したタスクを手続きと引数の組(アクティビティと呼ばれる)として実行待ちキューにつなぐ。軽量プロセスは未実行のアクティビティをキューから取って実行することを繰り返す。

2.2 「遺言」コンストラクトによる改良方式

しかしタスク間に待ち合わせがあるようなプログラムの場合、基本のアクティビティ方式では多数の軽量プロセスを生成することになる[2]。「遺言」方式では、親タスクは子タスクを生成した後、後処理を子タスクに「遺言」すると同時に終了する。親タスクを実行していた軽量プロセスは後処理のための内部状態を保持する必要がないのでただちに別のタスクを実行可能となり、新たな軽量プロセスを生成する必要がなくなる。

Implementation and Evaluation of Activity Based Parallel Execution Mechanisms on a Shared Memory Machine by MASAYA NAKAHATA, TAKESHI MOTOHASHI, YASUICHI NAKAYAMA, LEO NAGAMATSU, KOICHIRO DEGUCHI and IWAO MORISHITA (University of Tokyo).

子タスクの生成は、

`make_child(手続き, 引数)`

のように行なう。この時、家系記述子(Family Tree Descriptor、以下FTD)と呼ぶ構造体が作成され、実行すべき手き・引数が格納される。子のFTDは親のFTDを指すポイタを持ち、親のFTDには現在処理を完了していない子のが格納される。親タスクは全ての子タスクが完了した後に行すべき処理があれば、

`make_will(手続き, 引数)`

のように「遺言」を残して終了し、FTDの「遺言」の領に手続き・引数が格納される。

子タスクを処理していた軽量プロセスは処理が終わるとのFTDを参照して、自分の実行していたタスクが最後に理を完了した子に当たる場合はただちに親の「遺言」を実する。

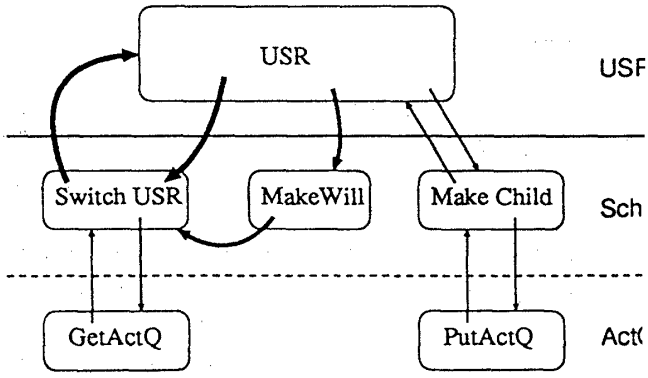

図1: アクティビティ方式の処理の流れ

2.3 アクティビティ方式の処理の流れ

図1にアクティビティ方式の処理の流れを示す。軽量プロスはキューからアクティビティーを取り(GetActQ)、実行る(USR)。実行中に子タスクが生成されると(MakeChilc キューにつながれる(PutActQ)。子タスクの完了待ちがときには、基本のアクティビティー方式の場合、親タス実行していた軽量プロセスは状態を保持したままサスペする。「遺言」方式の場合は「遺言」を残し(MakeWill)量プロセスを解放する。タスク切替(SwitchUSR)の時に量プロセスがアクティビティーを取ることを繰り返す。

3 実機上でのシステムの実現

3.1 ハードウェアの構成

SPARC の integer unit 、64KB の cache と MMU をプロセッサ・エレメント (PE) として用いた。PE-Memory 間は VMEbus のデータバスを 64bit 幅に拡張した共有バス構成とした。

プログラムのコンパイル、ロード、実行の制御及び I/O のために VMEbus-Sbus インターフェイスを介して SPARC station に接続した。

3.2 並列実行管理機構の移植

マルチプロセッサ用のプログラムを動作させるために必要な機能はソフトウェアで実現した。プロセッサの同期機構は、共有メモリ上にロックを用意し、SPARC の不可分メモリ操作命令を用いてロックを取得することにより実現した。fetch& add の機能はロックを取得したものだけがメモリを操作できるようにすることで実現し、プロセッサの休止と再開の操作はメモリ上のフラグをみて、busy wait する方式としたのでバスの負荷は重くなる。

これらのハードウェア及び同期機構の上に基本のアクティビティ方式と「遺言」方式の実行管理機構を移植した。

4 実験

実験には 1 台及び 2 台の PE に対して cache-off で、前述の基本方式 (ACT) と「遺言」方式 (WILL) 実行管理機構を用いて、7 都市巡回セールスマン問題 (枝刈りあり) を実行させた (図 2)。ユーザプログラム、システム、アイドルの各所要時間をプロセッサクロック単位で計測した。システムのうちアクティビティキュー操作コスト (ActQ) とアクティビティ生成・切替コスト (Sche) とを分けてある。

表 1に各作業のコストを示す。「遺言」方式で MakeWill として「遺言」を残すような場合に基本方式では子タスク終了まで軽量プロセスを中断して待つ (Suspend)。「遺言」方式では fork-join 型の問題を軽量プロセスの生成・切替なしに処理するため単位操作のコストが減少するので、システム時間が軽減する。また、基本方式では生成する軽量プロセス数が 213 であるのに対して、「遺言」方式ではプロセッサ数と同じ 2 であり、スタックの消費も少ない。

なお、台数効果が得られていないのは、プロセッサが複数台の時にはバスマスタ切替のコストがかかるためである。

5 まとめ

基本のアクティビティ方式及びその改良である「遺言」方式並列実行管理機構を簡単な共有バス型の並列機に実装し、

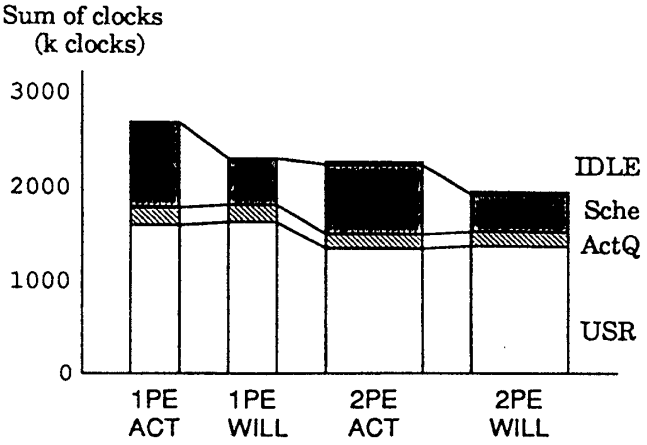

図 2: 基本方式と「遺言」方式の実行時間の比較

表 1: 各作業のコスト

作業	実行機構		
	基本方式	遺言方式	回数
PutActQ	269	277	249
GetActQ	308	308	249
MakeChild	439	440	452
Suspend	1683	-	1
MakeWill	-	202	1
SwitchUSR	2088	516	203

(単位: クロック数 / 回)

システムの各処理にかかる時間を計測することにより、プロセス生成・切替のコストが主要であり、「遺言」方式が有効であることを確認した。今後、より詳細な検討を行なう予定である。

謝辞

システム実現および評価に協力された田中玲子君、宮司卓佳君に感謝する。

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 (1993 掲載予定).