共有メモリ型並列機のためのアクティビティ方式並列実行機構(1)

——子プロセス待ちにおいて後処理を分割し性能改善を図る方式——

3P−1　中山泰一，白木光彦*，永松礼夫，森下巖
(東京大学工学部)

1．まえがき

共有メモリ型の汎用高並列計算機において、非常に多数の細粒度のタスク**を並列に実行するための並列実行管理機構として、あらかじめプロセッサの台数と同数の軽量プロセスを用意しておき、これらを繰り返し使用するアクティビティ方式が提唱されている[1]。その利点として、どのような形式の並列プログラムにも適用でき、タスクの実行中にサスペンドがまったく発生しなければ高い効率が実現できることが確認されている。

しかしながら、ネストしたfork-join形式の並列プログラムにおいて、親タスクによる子タスクの完了待ち合わせにより多数のサスペンドが発生し、従来のアクティビティ方式では顕著な効率の向上が得られない。すなわち、サスペンドが発生した場合には、プロセッサを有効に利用するために新たに軽量プロセスを生成する。これに必要なコストが小さくないからである。

上記形式のプログラムの実行効率をも向上させるためには、子タスク待ちにおいて軽量プロセス生成を行わないように工夫する改良と、子タスク待ちの場合の軽量プロセス生成・消滅コストを軽くする改良[3]とが考えられる。本発表では前者を試みるための、「遺言」という新しいコンストラクトを追加する方式について述べる。この方式に基づいた並列実行管理機構を試作し、シミュレーションによる実験を行った結果、アクティビティ方式の利点を活かしつつ、プロセッサ時間とメモリ消費量が大幅に節減できることが示された。

2．アクティビティ方式の基本原理と問題点

アクティビティ方式では、並列に実行すべきタスクの発生と、タスクの実行機構の用意を別個に取扱う。並列実行可能なタスクが発生すると同時にその実行のための軽量プロセスを生成することはしない。

応用プログラムでは、並列に実行させるタスクを1個の手続きとその引数の形（アクティビティとよぶ）で与えるものとする。これを管理機構は1本のキューに保持する。一方、あらかじめプロセッサ台数と同数用意された軽量プロセスは、未実行のアクティビティをキューから1個取ってきて実行し、それが完了すると次のアクティビティを取ってくる、を繰り返す。あらかじめ用意した軽量プロセスを繰り返し利用するので、軽量プロセスの生成・消滅に必要なプロセッサ時間、および、メモリ消費を節減できる。

(自分が実行する) 手続き
(〃)　引数
親のＦＴＤへのポインタ
「末っ子」のＦＴＤへのポインタ
現在生きている自分の子の数
「遺言」の手続き
〃　引数

図1　家系記述子(Family Tree Descriptor)の構造

しかしながら、軽量プロセスが同期のためにサスペンドすることが起きた場合、プロセッサを無駄なく利用するためには新しい軽量プロセスを生成して実行を開始させなければならない。並列アルゴリズムには、問題を同種の小問題に分割することを再帰的に行い、それぞれを並列実行させる型のもの（ネストしたfork-join形式）がしばしば用いられる。親タスクが再帰的に子タスクを生成していき、しかも、それぞれの親タスクがそのすべての子タスクの実行を待って後処理を実行する場合、親タスクによる子タスクの完了待ち合わせにより多数のサスペンドが発生し、従来のアクティビティ方式では多数の軽量プロセスを生成しなければならなくなる。これに必要なコストは小さくない。

3．「遺言」コンストラクトの導入[2]

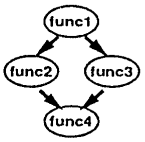
子タスク待ちにおいて無用の軽量プロセス生成を行わないようにするために、「遺言」という新しいコンストラクトを導入する。

具体的には、まず、タスクの親子関係を記述するために、家系記述子（ＦＴＤ）とよぶ構造体（図1）を新たに導入する。子タスクが生成される時点でＦＴＤを用意して、親のＦＴＤにポインタを用いてリンクする。これにより子タスクは自分の親が誰であるかを知ることが可能となる。また、親のＦＴＤには現在生きている自分の子の数を格納しておく。

Activity Based Parallel Execution Mechanism on Shared Memory Machines (1)
— A New Construct "Make a Will" for Post-processings to be Executed After the Completion of All the Child Tasks —
Yasuichi Nakayama, Mitsuhiko Shiraki*, Leo Nagamatsu and Iwao Morishita (University of Tokyo)

*現在、(株)日立製作所宇宙技術推進本部

**本稿では、並列プログラムにおいて並列に実行される単位を「タスク」とよぶ




```
func1()
{
    make_job(func2,arg2);
    make_job(func3,arg3);

    make_will(func4,arg4);
}
```

図２　子タスク func2,func3 の生成と、
「遺言」func4 作成のシステム・コール

次に、親タスクが後処理を実行しなければならない場合には、図２のように後処理を手続きと引数の形で、

make_will(後処理の手続き, 引数)

というシステム・コールを用いて宣言する。これによりＦＴＤの「遺言」の領域に後処理の手続きと引数の情報が格納され、同時に親タスクは「遺言」を遺したまま強制的に終了される。親タスクを実行していた軽量プロセスはその内部状態を保持しておく必要がないので、そのまま次のタスクの実行を開始できる。

作成した「遺言」は、子タスクのすべてが実行を完了した後ただちに実行させる方式とするのが、全処理時間の短縮に有利である。そこで、子タスクには処理の終了後、親のＦＴＤを参照して自分が子タスクのうちで最後に処理を終えたものであるか否かを判定させ、自分が最後に処理を終えた子の場合は、図３に示すように親のタスクの「遺言」をただちに実行する。その「遺言」を終えた結果、親が「親の親」から見て最後に処理を終える子に当たる場合には、さらに「親の親」の「遺言」もただちに実行する。

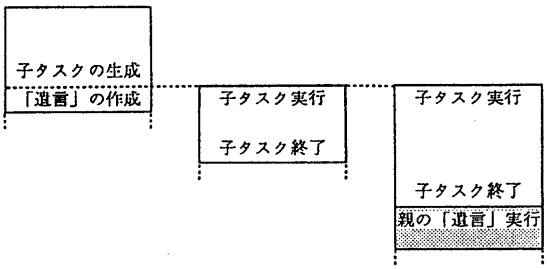


図３　「遺言」の作成と、その実行のタイミング

４．実験

「遺言」のコンストラクトを導入した並列実行機構の性能を評価するために、シミュレータによる実験を行った。並列機は、単一命令流プロセッサを直接理想共有メモリに結合したものとして、プロセッサ台数を８台とした。応用プログラムは、７都市巡回セールスマン問題を解く並列プログラムを用意した。

実験結果を図４、表１に示す。従来方式と比較して、処理時間は約15%削減され、スタック消費は約1/63ですむことが示された。

５．結論

「遺言」のコンストラクトの導入により、子タスク待ちにおいて軽量プロセス生成を行わないようにすることができ、アクティビティ方式の利点を活かしつつ、ネストしたfork-join形式のプログラムの実行効率をも向上できる。実験によりプロセッサ時間とメモリ消費量が大幅に節減できることが確かめられた。

また、「遺言」の形で表された後処理を、子タスクすべての実行の完了後に最優先で実行する方式としたことも、プロセッサのアイドル時間を削減させる効果があり、性能向上に役だっている。

より大規模な問題を解くにつれて、より実行効率の向上が実現されると予想される。特に、無用の軽量プロセス生成を行わないですむことによる、メモリ削減の効果が顕著に現れるものと思われる。

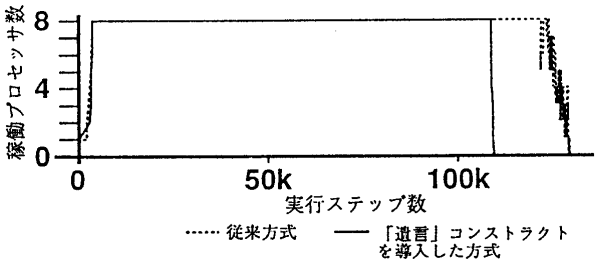


図４　７都市巡回セールスマン問題の並列処理状況と完了までの実行ステップ数

表１　７都市巡回セールスマン問題におけるスタック割り当てのためのメモリ消費（単位は bytes）

従来方式	105240
「遺言」コンストラクトを導入した方式	1664

参考文献

[1]田胡和哉，檜垣博章，森下巖：
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]中山泰一，永松礼夫，森下巖：
共有メモリ型並列機のためのアクティビティ方式並列実行機構の研究――タスクの親子関係を利用する後処理実行機構の導入――，情報処理学会オペレーティング・システム研究会報告，92-OS-55-3(1992).

[3]小林健一，中山泰一，永松礼夫，森下巖：
共有メモリ型並列機のためのアクティビティ方式並列実行機構(2) ――環境切替により後処理実行を行なう方式の提案――，
第45回情報処理学会全国大会，3P-2(1992).