2B－5

主記憶共有型マルチプロセッサシステム上でのマクロデータフロー処理の性能評価

松本 健　合田 憲人　岩崎 清　笠原 博徳
早稲田大学理工学部

1　はじめに

従来、マルチプロセッサシステム上でのFortranプログラムの粗粒度タスクの並列処理は、CRAYなどで見られるようなマルチタスキングによりOSあるいはランタイムライブラリを用いて実現されている[1][5]。マルチタスキングでは、複数のサブルーチン間の並列性をユーザーが並列性記述のための拡張言語を用いてソースプログラム上に明示する。従ってプログラムの並列性抽出はユーザーが行なわなければならず、プログラムを良く理解しているユーザしか並列性を十分に使用できない、大規模プログラムでは並列化のために長時間を必要とする、またマルチタスキングにおいてOSコールなどを用いた場合、実行時のオーバーヘッドが大きいという問題点がある。これに対し筆者らは、マクロデータフロー処理と呼ばれる、粗粒度タスク並列処理手法を提案し、マルチプロセッサシステムOSCAR上でその有効性を確認してきた。マクロデータフロー処理では、コンパイラが、基本ブロック、ループ、サブルーチンを粗粒度タスクとして定義し、それらの間の並列性を自動的に抽出した後、それらのマクロタスクを実行時にプロセッサあるいはプロセッサクラスタに低オーバーヘッドで割り当てるために、各Fortranソースプログラム専用のダイナミックスケジューリングコードを生成する。従ってマルチタスキングのようなOSコールを用いたスケジューリングと比べ、実行時のオーバヘッドを小さく抑えることができる。本稿ではこのマクロデータフロー処理手法を市販の主記憶共有型マルチプロセッサシステム上で実現した場合の性能について評価した結果を述べる。

2　マクロデータフロー処理手法

2.1　マクロタスクの生成

マクロデータフロー処理では、まずソースプログラムをマクロタスクと呼ぶ粗粒度タスクに分割する。マクロタスクには、単一基本ブロック、あるいは複数の基本ブロックを融合したブロック、または基本ブロックを複数に分割したブロックであるBPA、DOループあるいはバックワードブランチにより生成されるループすなわち最外側ナチュラルループであるRB、サブルーチンブロック（SB）の3種類がある。サブルーチンに関しては可能な限りインライン展開を行なうが、コード長等を考慮し効率的にインライン展開できない場合にそのサブルーチンをSBとして定義する。

2.2　マクロタスクグラフの生成

プログラムをBPA、RB、SBに分割した後、マクロタスク間の制御フロー、データフローを解析し、マクロフローグラフ[1]を生成する。次にマクロフローグラフからマクロタスク間の制御依存、データ依存を解析しマクロタスク間の最早実行可能条件を求め、それを最大の並列性を表現するマクロタスクグラフ[1][3]として表現する。

2.3　コード生成

各マクロタスクは、クリティカルパス法をダイナミックスケジューリング用に拡張したDynamic-CP法[6]によって、実行時にプロセッサに割り当てられる。スケジューリング時にＯＳコール等を行なうと命令実行時間の数百倍以上の処理時間[1]を要し、ダイナミックスケジューリングのオーバーヘッドが大きくなる。本手法ではコンパイラが各Fortranソースプログラムごとに生成するダイナミックスケジューリングコードを用いて実行時にマクロタスクのスケジューリングを行なうことによってオーバーヘッドを小さく抑えることが可能である。

3　実行方式

3.1　マクロタスクスケジューリング方式

今回利用する主記憶共有型マルチプロセッサシステム(Alliant FX/4)は4プロセッサしか持たないので、ここでは分散スケジューラ方式[4]を使用する。この方式では、図1スケジューリングコードが各PE上のプログラムコード上に埋め込まれており、各プロセッサにスケジューリング機能が分散される。各プロセッサは、割り当てられたマクロタスクの実行の前後にスケジューリングコードの実行を行なう。

このようなスケジューリング機能の実現においてはOSCARで使用したような集中スケジューラ方式[2]も考えられるが、今回対象とするFX/4のようにプロセッサ数が少ない場合、1PEをスケジューラとして専有するのは適切ではない。

3.2　マクロタスク実行方式

以下に、この分散スケジューリング方式の詳細について述べる。

まず最初に、1PEがマクロタスクの状態を管理するマクロタスク実行状態テーブル、マクロタスクの最早実行可能条件を管理するマクロタスク信号管理テーブル、さらに実行可能マクロタスクを貯めておくレディMTキュー

Performance evaluation of Macro-dataflow Computation on Shared Memory Multiprocessor System

Ken MATSUMOTO, Kento AIDA, Kiyoshi IWASAKI, Hironori KASAHARA
School of Science and Engineering, WASEDA UNIV.

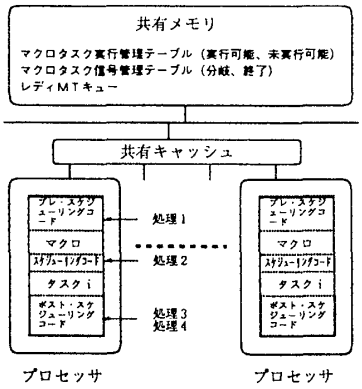

図 1: 主記憶共有メモリ上でのマクロデータフロー処理実現の概念図

を初期化後、プログラム開始時に実行可能なマクロタスクをレディMT キューに投入する。

以上の初期化終了後、各 PE は以下の動作をする。

1. レディMT キューから実行可能マクロタスクを取りだし、実行を開始する。
2. 実行中のマクロタスクに他のマクロタスクへの分岐が発生した場合は、信号管理テーブルに分岐情報を通達する。
3. マクロタスク実行終了後、信号管理テーブルに実行終了情報を通達し、未実行マクロタスクの最早実行可能条件を検査し、実行可能なマクロタスクがあればそのマクロタスクをレディMT キューへ投入する。
4. プログラム終了かどうかを検査し、終了でなければ 1 からの処理を繰り返す。

4 性能評価

4.1 評価方法

本節では Alliant FX/4 上でのマクロデータフロー処理の性能評価について述べる。マクロデータフローコンパイラが生成するスケジューリングコードを挿入されたプログラムコードを FX/C で作成した。分散スケジューラ型マクロデータフロー処理では、初期化処理終了後、全 PE が同一コードを実行するので、それを実現するために 1 回だけマルチタスキングを使用する。評価システムである Alliant FX/4 は、共有メモリを中心にして 4 台のプロセッサと、独立した I/O チャネルを備えた 12 個までの入出力専用のインタラクティブプロセッサを、共有バス・クロスバスイッチで結合された共有キャッシュメモリを用いて密結合した構成である。ピーク性能は、47.2MFLOPS、40MIPS である。

4.2 評価結果

ここでは対象帯状マトリックスを解く CG 法プログラムを用いて評価を行なった。このプログラムを使い、並列処理効果およびそのオーバーヘッドを評価するために、プログラム中の連立方程式の係数行列である配列のサイズを変化させて処理時間の測定を行なった。測定結果を表 1に示す。

表中の Seq. はシーケンシャル実行、size は CG プログラムで扱う配列の大きさ、time はプログラム実行時間

表 1: 評価結果 (スカラ)

size	Seq. time	Macro-dataflow time	Macro-dataflow ratio	Multitasking time	Multitasking ratio
16x16	0.2221	0.08638	2.571	15.45	0.01438
32x32	1.629	0.5150	3.163	29.26	0.05566
64x64	12.24	3.567	3.430	67.86	0.1803
128x128	98.24	27.49	3.574	173.2	0.5671
240x240	611.1	166.7	3.666	600.7	1.017

表 2: 評価結果 (ベクトル)

size	Seq. time	Mcaro-dataflow time	Mcaro-dataflow ratio	Multitasking time	Multitasking ratio
16x16	0.08434	0.04014	2.101	17.68	0.004770
32x32	0.6387	0.2077	3.075	34.69	0.01841
64x64	4.923	1.446	3.404	61.51	0.08004
128x128	39.54	11.62	3.403	154.0	0.2567
240x240	249.4	72.41	3.443	522.3	0.477

(単位秒) である。ratio はシーケンシャル実行時との速度比を表している。配列のサイズが大きくなればなるほどスケジューリングオーバーヘッドが小さくなり並列処理効果が得られていることが確認できる。また表よりマルチタスキングは、OS コールのオーバーヘッドのためにマクロデータフロー処理と比べ効率的な並列処理が行なえていないことがわかる。また表 2よりマクロデータフロー処理の速度比がスカラ時と比べ劣っているのは、ループのベクトル長が小さくなり十分なベクトル効果が上がらないためである。

5 まとめ

本稿では、Alliant FX/4 上でのマクロデータフロー処理の性能評価を行ない、マクロデータフロー処理が OSCAR のような分散共有メモリ型マルチプロセッサシステムだけでなく、市販の主記憶共有メモリ型マルチプロセッサでも有効であることを確認した。今後はマクロデータフローコンパイラを開発し、実マルチプロセッサ上での本手法の性能評価を行なう予定である。

参考文献

[1] 笠原：並列処理技術、コロナ社、１９９１.
[2] 本多，広田，笠原：階層型マルチプロセッサシステムＯＳＣＡＲ上での Fortran 並列処理手法，並列処理シンポジウム，JSPP'89, pp251-258（平 01-02）
[3] 本多，岩田，笠原：Fortran プログラム粗粒度タスク間の並列性検出手法，信学論 D-I, Vol.J73-D-I, No.12, pp951-960, (1990-12)
[4] 合田，松本，岡本，吉田，本多，笠原，成田：Fortran マクロデータフロー処理のマルチプロセッサスーパーコンピュータ上での評価，信学技報，CPSY-92-13，pp33-40, (1992-8)
[5] Alan H.Karp,Robert G.Babb II :A Comparison of 12 Parallel Fortran Dialects, IEEE Software (1988-9)
[6] 岩田，笠原：Fortran マクロタスクグラフのダイナミックマルチプロセッサスケジューリング手法、第３６回情報処理学会全国大会 3H-8,(1988-8)
[7] 本多，合田，岡本，笠原：Fortran プログラム粗粒度タスクの OSCAR における並列実行方式，信学論 D-I,Vol.J75-D-I,No.8,pp.526-535,(1992-8)