1B－8

商用共有メモリ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*

岩崎 清, 合田 憲人, 笠原 博徳, 成田 誠之助†
早稲田大学理工学部‡

1 はじめに

主記憶共有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並列処理手法として、従来よりマルチタスキングなどが用いられてきたが、マルチタスキングでは粗粒度タスク間の並列性抽出はユーザが行なわなければならない。また、実行時のタスクのダイナミックスケジューリングがOSコール等によって実現されているため、オーバーヘッドが大きいという問題があった[1]。このような問題に対処するために著者らは、従来よりマクロデーターフロー処理手法[2][3][4]を提案している。本手法では、コンパイラがプログラムの粗粒度タスク(マクロタスク)への分割、マクロタスク間の並列性抽出、ダイナミックスケジューリングコードの生成を自動的に行なうため、一般ユーザーでも簡単かつ低オーバーヘッドで粗粒度並列処理を行なうことができる。本稿では、このマクロデータフロー処理の性能評価を、商用共有メモリ型マルチプロセッサシステムKSR-1上で行なった結果について述べる。

2 マクロデータフロー処理手法

本章では、FORTRANプログラムのマクロデータフロー処理について概説する。

2.1 マクロタスク生成

はじめに、コンパイラはFORTRANプログラムをマクロタスク(MT)と呼ぶ粗粒度タスクに分割する。MTは、基本ブロック(BB)あるいは複数の基本ブロックからなるブロック(BPA)、繰り返しブロック(RB)、サブルーチンブロック(SB)から構成される。

2.2 マクロタスク間並列性抽出

MT生成後、MT間の制御フロー解析・データ依存解析を行ない、これらの情報を表現するマクロフローグラフを生成し、次にマクロフローグラフから、各MTの最早実行可能条件を求め、これを表現するマクロタスクグラフを生成する[2]。

2.3 マクロタスク分割・融合

マクロタスクグラフ生成後、MT間の並列性を向上させるためのMT分割、データ転送およびスケジューリングオーバーヘッドを最小化するためのMT融合を行なう[3]。

2.4 スケジューリングコード生成

マクロタスクグラフの情報をもとに、Dynamic-CP法[5]により実行時にMTをPEに割り当てるスケジューリングコードを生成する。一般的なOSコール等によるダイナミックスケジューリングは実行時のオーバーヘッドが大きいが、本手法ではコンパイラが生成したスケジューリングコードがMTのダイナミックスケジューリングを行なうため、オーバーヘッドを小さく抑えることができる。また、スケジューリング方式は、プログラムの並列性・使用するマシンの同期(排他制御)オーバーヘッド・プロセッサ台数・メモリ構成などを考慮して、スケジューリングコードの実行を単一PE上に集中させる集中スケジューラ方式または全PEに分散させる分散スケジューラ方式を使い分ける[4]。

2.5 階層型マクロデータフロー処理

プログラム全体が分割不可能な大規模ループで構成されている場合や、MT間に複雑なデータ依存・制御依存が存在し、単一階層マクロデータフロー処理ではMT間並列性を抽出できない場合には、プログラムの中のRB、SB内部でサブマクロタスクを生成しマクロデータフロー処理を階層的に適用することで、並列処理効果を向上させることができる[6]。

2.6 データローカライゼーション

複数のループに対してMT分割を適用した結果、生成されるループ間で配列変数要素間のデータ依存解析を行ない、データ転送の多いMTを同一PEに割り当て、MT間データ転送をローカルメモリを用いて行なうことにより、データ転送を減らすことができる[7]。

3 性能評価

本章では、KSR-1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。

3.1 KSR-1のアーキテクチャ

KSR-1は、32MBのローカルキャッシュを持つPEがリング状ネットワーク(ALLCACHE Engine)に接続されており、1PEのピーク性能は40MIPS, 40MFLOPSである。1リングあたりに接続できるPE数は32PEで、リングを階層的に接続することにより、最大1088PEまで接続可能である。

各PEのローカルキャッシュはALLCACHE Memory Systemによって全PEから参照・更新可能であり、論理的には共有メモリとして使用できる。ALLCACHE Memory Systemの概略図を図1に示す[8]。

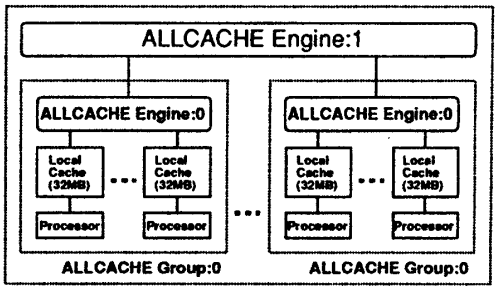

図1: KSR-1 ALLCACHE Memory System

3.2 CG法プログラムによる性能評価

3.2.1 CG法プログラム

本節では、対称帯状マトリクス係数行列を持つ連立方程式の解法であるCG(Conjugate Gradient)法プログラムを用いたマクロデータフロー処理の性能評価について述べる。このプログラムは、変数の初期化部分と実際に連立方程式の係数行列を解く収束計算ループから構成されているが、実行時間のほとんどが収束計算ループによって占められ、また、収束計算ループ内のサブマクロタスク間の並列性が得られるので、収束計算ループ内のサブマクロタスクに対して階層型マクロデータフロー処理を適用する。また、サブマクロタスク間のDo allループの並列性を利用するためサブマクロタスク(サブRB)に対してMT分割を適用する。収束計算ループ内のサブマクロタスクグラフを図2に示す。

---

*Performance Evaluation of Macro-dataflow Computation on Shared-memory Multi Processor System

†Kiyoshi IWASAKI, Kento AIDA, Hironori KASAHARA, Seinosuke NARITA
‡Waseda University

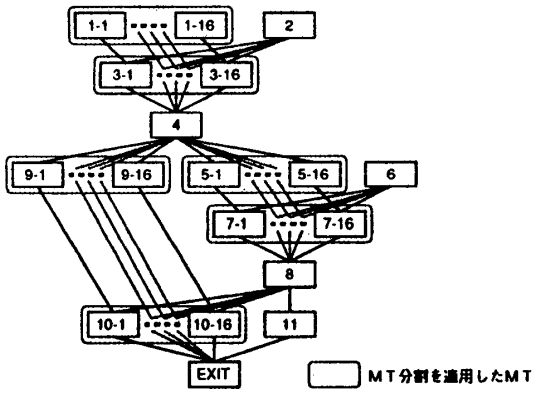

図 2: CG 法の収束計算ループ内マクロタスクグラフ

3.2.2 評価結果

KSR-1 上で 16PE(集中スケジューラ方式は 15+1PE) 使用した場合の並列処理による実行時間と実行速度向上比を表 1 に示す。

集中スケジューラ方式では、スケジューリング専用の PE(コントロールプロセッサ:CP) が各 PE に MT を割り当て、各 PE は割り当てられた MT の実行が終ると実行終了・分岐フラグを CP に送信し、そのフラグを受信した CP が実行可能になっていない MT の実行可能条件のチェックを行ない、実行可能 MT を各 PE に割り当てる。分散スケジューラ方式では、実行可能な MT の入ったキュー (レディMT キュー) から MT の取り出し、その MT の実行、実行可能になっていない MT の実行可能条件のチェック・レディMT キューへの挿入を、各 PE がそれぞれ行なう。このうち MT のキューへの挿入と取り出しは各 PE で排他的に行なわれる。

表 1: CG 法プログラムの実行結果

	係数行列のサイズ			
	128×128		256×256	
	時間 (秒)	速度比	時間 (秒)	速度比
Sequential	44.082	1.000	328.240	1.000
分散	5.780	7.626	30.457	10.777
分散,local	6.063	7.271	21.954	14.952
集中	5.657	7.793	32.620	10.063
集中,local	4.344	10.147	22.277	14.734
M-thread	35.159	1.253	101.387	3.237

集中:集中スケジューラ方式
分散:分散スケジューラ方式
local:データローカライゼーション適用
M-thread:マルチスレッディング

これらの結果より、マクロデータフロー処理を適用した場合の実行時間が、係数行列のサイズが 256×256、分散・local の時で、シーケンシャル実行時間の 1/14.95 倍と高い並列処理性能を得られることがわかる。

分散・集中スケジューラ方式の比較では、集中スケジューラ方式は、スケジューリング処理をスケジューリング専用の PE が行いスケジューリング管理変数の排他制御などを行なう必要がない分オーバーヘッドが少ないが、実際に計算を行なう PE が分散スケジューラより 1PE 少ないので、MT の処理時間が小さい場合 (係数行列のサイズが 128×128 の場合) では、スケジューリングオーバーヘッドの小さい集中スケジューラ方式の方が実行時間が短縮され、MT の処理時間が大きい場合 (係数行列のサイズが 256×256 の場合) では、実際に計算を行なう PE が 1PE 多い分散スケジューラ方式の方が実行時間が短縮されている。

データローカライゼーションの有無の比較では、行列サイズ 256×256 の時でデータローカライゼーションを適用した場合の実行時間が、適用しない場合の実行時間の 1/1.46 倍 (集中)、1.39 倍 (分散) になるなど、実行時間を短縮できることが確認された。ただし、分散スケジューラ方式で MT の処理時間が小さい場合 (係数行列のサイズが 128×128 の場合)、データローカライゼーションの処理が逆にオーバーヘッドとなりデータローカライゼーションを適用しない時より遅くなる場合があった。

一方、分割した MT に対して、スレッドの生成・生成したスレッドそれぞれで分割した MT の並列実行・スレッドのジョインを行なうことにより、並列処理を行なうマルチスレッディングを適用した場合は、係数行列のサイズが 128×128 の時で、シーケンシャル実行時間の 1/1.25 倍、256×256 の時で 1/3.23 倍と低い並列処理性能しか得られなかった。これは実行時のスケジューリングオーバーヘッドが大きいためである。これに対しマクロデータフロー処理では、係数行列のサイズが 256×256 の時でシーケンシャル実行時間の 1/14.95 倍であり、マクロデータフロー処理が低オーバーヘッドで効率良い並列処理を行なうことができることが確認された。

4 まとめ

本稿では、商用共有メモリ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について述べた。

マルチスレッディングとの比較では、マルチスレッディングはスケジューリングオーバーヘッドが大きく並列処理性能が低いのに対し、マクロデータフロー処理は低オーバーヘッドで効率良い並列処理を行なうことができることが確認された。

マクロデータフロー処理のスケジューリング方式では、集中スケジューラ方式は、スケジューリング処理をスケジューリング専用の PE が行うので共有スケジューリングデータのロック等に要するオーバーヘッドが少ないという利点があり、分散スケジューラ方式は、実際に計算を行なう PE を集中スケジューラ方式よりも 1PE 多く使用できるという利点があり、プログラムの並列性・タスクグレイン・マシンの同期性能に応じてプログラム毎にコンパイラがどちらの方式を選択するか決定することが今後重要となる。今後は、さらに実アプリケーションプログラムに対して性能評価を行なうとともに、VPP-500 や Cenju-3 など他の商用マルチプロセッサシステム上でも性能評価を行なう予定である。

なお、本研究にあたり KSR-1 を使用させて頂いたキャノンスーパーコンピューティング S.I.(株) の皆様に感謝致します。また、本研究の一部は文部省科学研究費 (一般研究 (b)05452354, 一般研究 (c)05680284) により行なわれた。

参考文献

[1] A.H.Karp, R.G.Babb II : A Comparison of 12 Parallel Fortran Dialects, IEEE Software Vol.5 no.5, pp52-67, (Sep.1988)

[2] 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 Vol.J73-D-I, No.12, pp951-960, (1990-12)

[3] 笠原, 合田, 吉田, 岡本, 本多:Fortran マクロデータフロー処理のマクロタスク生成手法, 信学論 Vol.J75-D-I No.8,pp511-525,(1992-08)

[4] 合田, 岩崎, 松本, 岡本, 笠原, 成田:主記憶共有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, 情処研報 ARC105-9, (1994-3)

[5] 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 Vol.J75-D-I No.8,pp526-535,(1992-08)

[6] 岡本, 合田, 宮沢, 本多, 笠原:OSCAR マルチグレイニンコンパイラにおける階層型マクロデータフロー処理, 情処学論, Vol.35, No.4, pp.513-521,(1994-4)

[7] 吉田, 前田, 尾形, 笠原:Fortran 粗粒度並列処理における Doall/シーケンシャルループ間データローカライゼーション手法, 信学論, Vol.J78-D-I, No.2, (1995-2)

[8] KENDALL SQUARE RESERCH TECHINICAL SUMMARY, Kendall Square Research Corporation, (1992)