2B－3 マルチグレイン並列処理における タスク融合を用いたデータローカライゼーション手法*

前田誠司，吉田 明正，笠原博徳†

早稲田大学 理工学部‡

1 はじめに

マルチプロセッサシステム上における Fortran プログラムの自動並列処理では、従来 Doall、Doacross 等のループ並列化が用いられている [1][2]。しかし、ループ並列化ではループ以外の部分の並列性を抽出することができないという問題があった。この問題点を解決するために、筆者らはステートメント間の近細粒度並列処理、ループのイタレーション間の中粒度並列処理、サブルーチン・ループ・基本ブロック間の粗粒度並列処理を階層的に組み合わせ、プログラム全域の並列性を利用するマルチグレイン並列処理をすでに提案している [2][3]。本稿では、このマルチグレイン並列処理において、各階層のタスク間データ転送オーバーヘッドを軽減するための、タスク融合を用いたデータローカライゼーション手法を提案する。また、提案手法を用いたコンパイラは OSCAR 上でインプリメントされており、本稿ではその性能評価についても述べる。

2 マルチグレイン並列処理

本マルチグレイン並列処理手法では、Fortran プログラムを階層的に粗粒度タスク・中粒度タスク・近細粒度タスクに分割し、各々の粒度で並列処理を行なう。

2.1 粗粒度並列処理

本粗粒度並列処理手法は、マクロデータフロー処理 [2][4][5] と呼ばれ、まず最初にコンパイラは、Fortran プログラムをサブルーチンブロック (SB)、繰り返しブロック (RB)、擬似代入ブロック (BPA) 等のマクロタスクと呼ぶ粗粒度タスクに分割する。次にコンパイラは、マクロタスク間の制御フロー及びデータフローを解析し、マクロフローグラフ (MFG) を生成後、各マクロタスクの最大の並列性を表す最早実行可能条件を求める。

この最早実行可能条件を基にコンパイラは、条件分岐等の実行時非決定性に対処し、さらにスケジューリングオーバーヘッドを最小化するために、実行時に各マクロタスクをプロセッサクラスタ (PC) に割り当てるダイナミックスケジューリングルーチンを生成する。このダイナミックスケジューリングは粗粒度タスクに対して適応されるため、オーバーヘッドは相対的に小さく抑えられる。

2.2 中粒度並列処理

2.1 で述べたように、マクロデータフロー処理によりマクロタスクは実行時に PC に割り当てられる。PC に割り当てられたマクロタスクが Doall ループまたはリダクションループ (総和計算ループ) で構成されている場合、それらのマクロタスクは PC 内部のプロセッサエレメント (PE) によって中粒度で並列処理される。

2.3 近細粒度並列処理

PC に割り当てられたマクロタスクが BPA である場合、マクロタスクは 1 ステートメントからなる近細粒度タスクに分割される。次にそれらの近細粒度タスクは、データ転送を考慮したヒューリスティックアルゴリズム CP/DT/MISF 法 [2] により、PC 内部の PE にコンパイル時に割り当てられ並列処理される。

3 データローカライゼーション手法

本節では、マルチグレイン並列処理において共有メモリを介したデータ転送オーバヘッドを軽減するためのデータローカライゼーション手法を提案する。ここでデータローカライゼーションとは、同一プロセッサに割り当てられたマクロタスク間では、共有メモリを介さず、ローカルメモリを介してデータ授受を行なう手法である。本手法は、(1) ループ整合分割、(2) マクロタスク融合、(3) 融合 MT 内でのデータローカライゼーション用データ転送コード生成により実現される。

3.1 ループ整合分割

マクロデータフロー処理では、大規模 Doall ループが単一マクロタスクとして 1 つのプロセッサクラスタ (PC) だけに割り当てられるのを防ぐため、Doall ループを PC 台数あるいはその倍数個のマクロタスクに分割し、それらを複数 PC 上で並列実行する。しかし、各 Doall ループを他の Doall ループ (MTG 上のデータ依存先行・後続タスク) におけるデータ使用・定義範囲を考慮せずに独立に分割すると、他の Doall ループとの間でデータ定義・使用の局所化を行ないローカルメモリを介したデータ授受を行なうためのマクロタスク融合を適用できなくなる可能性がある。

そこで、MTG 上で単一のデータ依存エッジにより接続された Doall ループ (マクロタスク) を分割するときには、分割により生成される部分ループ間での配列変数の使用範囲が等しくなるように整合分割 [6] を行なう。

3.2 マクロタスク融合

本手法では、前述のように、ループ整合分割後にマクロタスク間に大きいデータ転送が存在するマクロタスク集合を融合し、実行時のダイナミックスケジューリングではこの融合マクロタスクを 1 つのマクロタスクのように扱い PC に割り当てる方法をとる。このコンパイル時のマクロタスク融合により、データ転送量の多いマクロタスク集合は実行時に低オーバヘッドで同一 PC に割り当てられる。

---

*A Data-Localization Scheme for Multi-Grain Parallel Computation Using Task-Fusion

†Seiji MAEDA, Akimasa YOSHIDA, Hironori KASAHARA

‡Waseda University

3.3 データローカライゼーションの実行方式

本手法では、3.2 で述べた方法で生成された融合マクロタスクに、データローカライゼーション手法を適用する。本データローカライゼーション手法を用いることにより、融合されたマクロタスク内での共有メモリアクセスを最小化できるため、データ転送オーバーヘッドを軽減することができる。

3.3.1 融合マクロタスク内データ転送解析

融合により生成されたマクロタスク内で、データローカライゼーションのためのデータ依存解析を行なう場合、融合マクロタスク内の並列処理手法が、中粒度並列処理と近細粒度並列処理の 2 通り可能であるので、これらのタスク間のデータ転送を考慮しなければならない。

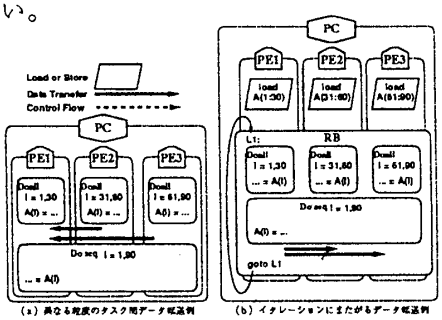

図 1: データ転送の例

例えば図 1 (a) の様に、融合マクロタスク内に Doall ループとシーケンシャルループが混在している時に、Doall ループのイタレーションが各 PE に分割され、またシーケンシャルループの内部ステートメントが近細粒度並列処理される場合、配列 A を参照するタスクが PE1 に割り当てられる時には、図中の矢印で示される様に、異なる粒度で並列処理されるタスク間データ転送を行なう。

また、図 1 (b) の様に、RB 内に、Doall ループとシーケンシャルループが混在している時に、シーケンシャルループ内で PE1 に割り当てられた近細粒度タスクによって定義された配列 A を RB の後続イタレーションにおいて各 PE 上の Doall ループが参照する場合には、図中の矢印で示される様に、RB のイタレーションの最後に、PE1 から各 PE にデータを分散させるためのタスク間データ転送を行なう。

これら PE 間とイタレーション間の双方に関係しているデータ転送について解析を行ない、データ転送命令を生成する。

3.3.2 融合マクロタスクデータ転送

ローカライズを適用する配列変数は、実行時にローカルメモリのデータにアクセスするようにコード生成される。したがって、その配列変数への参照が発生するまでの間に、参照する範囲のデータはローカルメモリ上に置かれていなければならない。このため、当該プロセッサ上で参照前に定義されない範囲のデータは、図 1 (b) 上部の load に示す様に融合マクロタスク実行前に共有メモリからローカルメモリに転送されなければならない。

3.3.3 データ転送時間の予測とデータローカライゼーション用コード生成

融合マクロタスク内のデータ依存形状を考慮せず、すべての配列変数に対してローカライゼーションの適用を行なうと、共有メモリに直接アクセスする場合に比べて、データ転送のオーバーヘッドが大きくなり過ぎて、逆に融合マクロタスクの処理時間が延びてしまうことがある。そこで、本手法では、各配列変数ごとにローカライズする場合としない場合の融合マクロタスク内でのデータ転送時間を求め、ローカライゼーションによってデータ転送時間の短縮される可能性が高い配列変数だけにローカライゼーションを適用する。ローカライゼーションの適用されない配列変数は、共有メモリに直接アクセスすることになる。

4 OSCAR 上での性能評価

OSCAR[7] は、ローカルメモリと分散共有メモリを持つ 32 ビット RISC プロセッサ 16 台を、集中共有メモリに 3 本のバスで接続した共有メモリ型マルチプロセッサシステムである。OSCAR は、PE を平等に結合したアーキテクチャとなっているが、複数の PE をソフトウエア的にクラスタ化することにより、複数の PC を効率よくシミュレートすることができる。ここでは、性能評価のプログラムとして非対称係数行列を持つ連立 1 次方程式間接解法である CGS 法プログラムを用いる。このプログラムを OSCAR シミュレータ上で 1PE を用いてシーケンシャル実行すると、処理時間は 108.0[ms] であったのに対し、提案する整合分割を用いると 2PC・3PE(計 6PE) で 24.8[ms] の処理時間が得られた。さらに、マクロタスクを融合しデータローカライゼーションを適用すると、共有メモリアクセスに伴うデータ転送オーバーヘッドが軽減されるため、2PC・3PE(計 6PE) で処理時間は 19.2[ms] となり、シーケンシャル処理と比べて 5.63 倍の速度向上が得られる。

5 むすび

本稿では、マルチグレイン並列処理においてタスク融合を用いたマクロタスク内でのデータローカライゼーション手法について述べ、その有効性を OSCAR シミュレータ上で確かめた。今後の課題としては、スタティックスケジューリングを用いたマクロデータフロー処理と組み合わせ、より広範囲かつ高精度なデータローカライゼーションを実現することなどがあげられる。

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] D.A.Padua, M.J.Wolfe: "Advanced Compiler Optimizations for Super Computers", C.ACM, **29**, 12, pp.1184-1201 (Dec.1986).

[2] 笠原:"並列処理技術", コロナ社 (1991-06).

[3] H.Kasahara, H.Honda, A.Mogi, A.Ogura, K.Fujiwara, S.Narita: "Multi-Grain Parallelizing Compilation Scheme for OSCAR", Proc. 4th Workshop on Language and Compiler for Parallel Computing, 19 (1991).

[4] 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), **J73-D-I**, 12, pp951-960 (1990-12).

[5] 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 (D-I), **J75-D-I**, 8, pp.511-525 (1992-08).

[6] 吉田, 前田, 尾形, 岡本, 本多, 笠原: "マクロデータフロー処理におけるデータローカライゼーション手法", 信学技法, CPSY93-23 (1993-08).

[7] 笠原, 成田, 橋本: "OSCAR のアーキテクチャ", 信学論 (D), **J71-D**, 8 (1988-08).