2E－5　一般的なマクロタスクグラフに対するループ間データローカライゼーション手法

松崎 秀則†, 吉田 明正‡, 岡本 雅巳††, 松井 巌徹†,
小幡 元樹†, 宇治川 泰史†, 笠原 博徳†

†早稲田大学理工学部電気電子情報工学科,
‡東邦大学理学部情報科学科, ††(株) 東芝

1　はじめに

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階層型マクロデータフロー処理 [5] が提案されている。この階層型マクロデータフロー処理において、ループやサブルーチン等の粗粒度タスク (マクロタスク) のプロセッサへの割当がダイナミックスケジューリング [4] によって行なわれる場合、マクロタスク間で発生するデータ転送は一般的に集中共有メモリを介した方式がとられている [5]。しかし、このような方式では集中共有メモリへのアクセスが多く、データ転送オーバーヘッドが大きくなるという問題が生じる。そこで、マクロタスク間のデータ転送オーバヘッドを軽減すると同時に負荷を均等にして効率的な並列処理を行なうために、各ループでの配列データの使用範囲を考慮して手順とデータを分割し、さらにデータ転送量の多い複数のマクロタスクを同一プロセッサに割り当てローカルメモリを介したデータの授受を行なうためのループ間データローカライゼーション手法を提案する。

2　階層型マクロデータフロー処理

本章では階層型マクロデータフロー処理手法について述べる。OSCARマルチグレインコンパイラ [2, 4, 5] ではプログラムを以下に示す三種類のMTに階層的に分割する。

- BPA(Block of Pseudo Assignment statements)
- RB(Repetition Block)
- SB(Subroutine Block)

階層的にMTを生成した後、各階層でMT間のコントロールフロー解析、およびデータフロー解析を行う。解析したコントロールフローとデータフローは各階層毎に生成するマクロフローグラフで表現する。次に各階層の (サブ) マクロフローグラフに対して最早実行可能条件解析 [3] を行いMT間の並列性を抽出し、それをもとにマクロタスクグラフを生成する。

3　データローカライゼーション手法

本章では、階層型マクロデータフロー処理がダイナミックスケジューリングで実行される場合におけるデータローカライゼーション手法について提案する。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MT間では共有メモリを介さずローカルメモリを介したデータ授受を行えるように、ループを分割しスケジューリングする手法である。本手法は、(1) ループ整合分割、(2) データ転送の解析、(3) データ転送時間の予測とコード生成、(4) パーシャルスタティックタスク割当を伴うダイナミックスケジューリングルーチン生成により実現される。

3.1　ループ整合分割の拡張

本手法では、データ依存関係のあるRBを配列データの使用範囲が等しくなるようにループ整合分割 [6, 7, 8] を行う。従来のループ整合分割は既にOSCARマルチグレインコンパイラに以下の方法で実装されている。

1. MTG上で単一のデータ依存エッジで接続されたRB集合をターゲットループグループ (TLG) とする。
2. TLG内であるRBのイタレーションと他のRBのイタレーションとの間のループ間データ依存を解析する。
3. ループ間データ依存解析の結果に基づいて、各RBをデータの使用範囲が等しくなるように整合分割する。

しかし、複数のRBで生成した配列データを他のRB中で一括して使用する場合等には、1つのRBから複数のRBへデータ依存エッジが張られるので、1. のように1つのエッジで結ばれたRBの組をTLGとして選択しなくてはならない。そのため選択されなかったRBを同じTLGに含めることが出来ず、それらの間でのデータの授受は集中共有メモリを介したものとなってしまう。

そこで、本稿では一般的な形状のマクロタスクグラフにおいて共通の配列データを参照するRBを可能な限り同一のTLGに含め、集中共有メモリを介したRB間のデータ転送を最小限に抑えるための手法を提案する。

3.1.1　ループ整合分割の対象となるRB集合 (TLG)

以下の条件を満たすRBをループ整合分割の対象となるRB集合 (TLG) とする。

- Doall、Reduction、Sequentialループである。
- 実行確定条件 (制御依存) が同じである。
- TLG中のRBと配列変数データ依存で結ばれている。(トランシティブなデータ依存エッジも考慮)
- 各RBでの分割対象となる配列において、同じ次元の添字式に最外側ループの誘導変数が使用されている。またその添字式は誘導変数の一次式である。

この条件から抽出されるRBの集合は同じ条件の下で実行されその中に条件分岐は含まれない。

3.1.2　TLG内のループ間データ依存解析

3.1.1で抽出したTLGに対して、次の手順でTLG内ループ間データ依存解析を行なう。

1. ダミーの標準ループ (DSL) を作り、RB集合のうち同じTLG内に後続RBを持たない全てのRBからのデータ依存エッジを張る。
2. 標準イタレーション変換係数 (SIC) を計算する。ただしDSLのSICは1とする。以後の手順ではSICをもとに各RBをDSLのインデクスに換算したものを用いて計算を行なう。
3. 直接データ依存エッジで結ばれているRB間の相対的なイタレーションに関するデータ依存情報 (DirILD) を計算する [8]。例えば RB_j のある t 番目のイタレーションを計算する為に、RB_i の t 番目から $(t+1)$ 番目までイタレーションが必要な場合、$DirILD(RB_i, RB_j) = [0:1]$

---

* A Data-Localization Scheme among Loops in General Macrotask-Graph

Hidenori MATSUZAKI†, Akimasa YOSHIDA‡, Masami OKAMOTO††,Gantetsu MATSUI†, Motoki OBATA†, Yasushi UJIGAWA†, Hironori KASAHARA†

† Waseda University. 3-4-1 Ohkubo Shinjuku-Ku, Tokyo 169, ‡Toho University, ††Toshiba Corporation

の様に表される。ここで、各RBからDSLへのデータ依存エッジに関するDirILDは全て[0:0]とする。

4. TLG内のタスクグラフにおいて、後続RBとのDirILDおよび後続RBのILDから、対象RBとDSL間のイタレーションに関するデータ依存情報(ILD)を計算する[8]。例えばRB_i、RB_j、RB_kの順に直列にデータ依存を持つグラフを考えてみると、RB_iとRB_kの関係は以下の式によって計算される。式中の$_{L,U}$はそれぞれ依存範囲の下限値、上限値を表している。

$$ILD_L(RB_i, RB_k) = ILD_L(RB_j, RB_k) + DirILD_L(RB_i, RB_j)$$
$$ILD_U(RB_i, RB_k) = ILD_U(RB_j, RB_k) + DirILD_U(RB_i, RB_j)$$

また複数の後続RBがある場合は、各後続タスクからILDを計算して最も依存範囲の大きいILDの上限値に他の後続タスクから計算したILDの上限値を合わせるように、DSLとのDirILDを調整する。これにより3.1.3のループ分割を行った際に発生する他プロセッサへのデータ転送量を最小に抑えることができる。

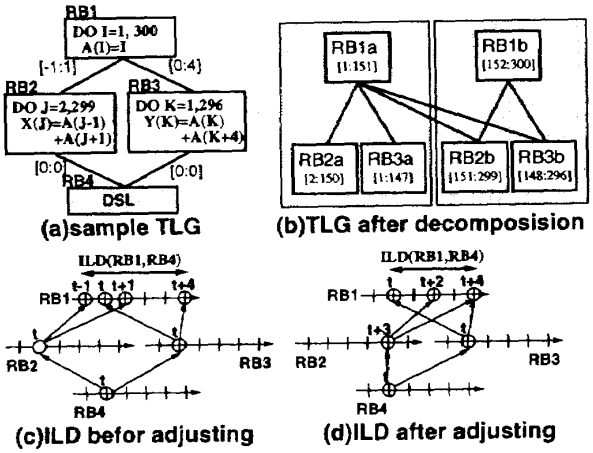

図 1: ループ整合分割の例

例としてRB_1,RB_2,RB_3からなる図1(a)のTLGについて依存解析をおこなう。まずDSLとしてRB_4を作り、次に添字式からDirILDは$DirILD(RB_1, RB_2) = [-1 : 1]$,$DirILD(RB_1, RB_3) = [0 : 4]$, $DirILD(RB_2, RB_4) = [0 : 0]$,$DirILD(RB_3, RB_4) = [0 : 0]$となる(図1(a))。最後に各RB-DSL間のILDを計算する。RB_2、RB_3は直接DSLに依存エッジがあるためDirILDをそのまま使い、$ILD(RB_2, RB_4) = [0 : 0]$,$ILD(RB_3, RB_4) = [0 : 0]$となる。$RB_1$は後続RBとして$RB_2$と$RB_3$を持っているので、各後続RBからのILDを計算すると、$ILD^{viaRB_2}(RB_1, RB_4) = [-1 : 1]$,$ILD^{viaRB_3}(RB_1, RB_4) = [0 : 4]$となる(図1(c))。ここで依存範囲は$\|ILD^{viaRB_2}(RB_1, RB_4)\| = 2$, $\|ILD^{viaRB_3}(RB_1, RB_4)\| = 4$であり、$RB_3$からのILDの方が依存範囲が大きいので、こちらを$RB_1$のILDとし、$ILD^{viaRB_2}(RB_1, RB_4)$の上限値をこの上限値に合わせるために、$DirILD(RB_2, RB_4) = [3 : 3]$とする。この結果$RB_2$のILDも変化して$ILD(RB_2, RB_4) = [3 : 3]$となる(図1(d))。

3.1.3 TLG内のループ分割

$ILD_L(RB_i, DSL)$,$ILD_U(RB_i, DSL)$をそれぞれ標準イタレーションデータ依存範囲の下限値(SIL_i)、上限値(SIU_i)とすることにより、従来より提案されている整合分割方法[8]を用いてループ分割を行なう。なおこの分割においてデータ転送量の多い部分ループ集合は、バーシャルスタティックタスク割り当てを用いたダイナミックスケジューリングルーチンにより同一プロセッサに割り当てられ、ローカルメモリ経由でデータの授受が行なわれる[8]。この際、分割する際に生じる共通データ定義ループについては、インデックスが小さい方の部分ループに含める。この分割方法にもとづき図1(a)は図1(b)のように分割される。

4 OSCAR上での性能評価

本章では、提案手法をアプリケーションプログラムに適用し、OSCARシミュレータ上で評価した結果について述べる。

性能評価プログラムとして、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法(Conjugate Gradient Method)のデータ初期化部分を用いた。結果は表1に示すように、1プロセッサに対する4プロセッサでの実行時間はDoall処理が3.71倍、またマクロデータフロー処理をダイナミックスケジューリングで実行した場合、従来手法の整合分割・データローカライゼーション手法で4.40倍の速度向上が得られた。さらに今回提案した整合分割・データローカライゼーション手法では4.92倍となり、従来手法と比べて11.6%速度が向上している。以上のように整合分割・データローカライゼーションによる大幅な処理時間の短縮が確認できる。以上の結果でスーパーリニアスピードアップが得られているのは、1プロセッサでの実行においてデータサイズがローカルメモリサイズより大きいために、集中共有メモリを使用しているためである。

表 1: CG法初期化ルーチンでの性能評価

PE数	実行方式	時間 [msec]	1PE比
1	Sequential処理	19.17	1
4	Doall処理	5.16	1/3.71
4	マクロデータフロー処理(従来の整合分割・ローカライズ)	4.35	1/4.40
4	マクロデータフロー処理(本手法による整合分割・ローカライズ)	3.90	1/4.92

5 まとめ

本稿ではデータ依存のみを持つマクロタスクグラフあるいは部分タスクグラフにおいて、配列データを共有する複数のループ間のローカルメモリを介したデータの授受を効果的に行なうためのデータローカライゼーション手法を提案した。また、OSCARシミュレータ上でのCG法初期化ルーチンによる性能評価の結果により、従来のデータローカライゼーション手法に比べて11.6%の速度向上が得られ、提案手法の有効性が確認できた。今後の課題として条件分岐を含んだTLGの抽出とその依存解析によるより広範囲なデータローカライゼーションの実現があげられる。

本研究の一部は通産省次世代情報処理基礎技術開発事業マルチプロセッサコンピューティング領域研究により行なわれた。

参考文献

[1] 笠原: "並列処理技術", コロナ社 (1991-06).
[2] H.Kasahara, H.Honda, S.Narita: "A Multi-Grain Parallelising Compilation Scheme for OSCAR", Proc. 4th Workshop on Languages and Compilers for Parallel Computing (Aug. 1991).
[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, J73-D-I(12) (1990-12).
[4] H.Kasahara, H.Honda, M.Iwata, M.Hirota: "A Compilation Scheme for Macro-dataflow Computation on Hierarchical Multiprocessor Systems", Inter. Conf. on Parallel Processing (Aug. 1990).
[5] 岡本, 合田, 宮沢, 本多, 笠原: "OSCARマルチグレイ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(4) (1994-4).
[6] 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(2) (1995-02).
[7] 吉田, 前田, 尾形, 笠原: "Fortran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", 情処学論, Vol.35(9) (1994-9).
[8] A.Yoshida, K.Koshisuka, H.Kasahara: "Data-Localisation for Fortran Macro-Dataflow Computation Using Partial Static Task Assignment", ICS(1996)
[9] 笠原, 本多, 橋本: "OSCARのアーキテクチャ", 信学論 D, Vol.38,No1, (1989-1).