1L－5

階層型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*

越塚 健一†, 吉田 明正†, 岡本 雅巳†, 笠原 博徳†

†早稲田大学理工学部電気電子情報工学科

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をダイナミックスケジューリング[4]で実行する場合、ループやサブルーチンなどの粗粒度タスク(マクロタスク)が実行時にプロセッサに割り当てられるため、マクロタスク間で発生するデータ転送を集中共有メモリを介して行う方式がとられていた[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。本稿では、この問題点を解決するために、階層型マクロデータフロー処理において、それらの間で多量のデータ転送を生じる可能性がある複数のマクロタスクを同一プロセッサに割り当ててローカルメモリ経由でデータ授受を行う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マクロタスク(MT)に階層的に分割する(図1)。

- BPA(Block of Pseudo Assignment statements)
  基本ブロックおよび複数の小基本ブロックを融合したブロック
- RB(Repetition Block)
  最外側ナチュラルループ
- SB(Subroutine Block)
  インライン展開が有効に適用できないサブルーチン

各階層のMTは階層的に定義されたプロセッサクラスタ(PC)間で並列処理される。また、各階層でPCに割り当てられたMTは、さらにPC内のプロセッサエレメント(PE)により階層的に並列処理される。例えばMTがBPAならば、BPA内部ではステートメントレベルの近細粒度並列処理を適用する。RBならばDo-allやDo-across などの中粒度並列処理、あるいはループボディの近細粒度並列処理、またRBが大規模であり内部でサブMTが階層的に定義できる場合には階層的にマクロデータフロー処理を適用する。この場合、サブMTはPC内で定義されるサブPCに割り当てられる。

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間の並列性を抽出する。この結果得られた各階層における各MTの最早実行可能条件をグラフで表現したマクロタスクグラフ(MTG)を各階層ごとに生成する。各階層のMTは、マクロタスクグラフに実行時不確定性(条件分岐等)が存在する場合はダイナミックスケジューリング、それ以外の場合にはスタティックスケジューリング[1]で実行される。

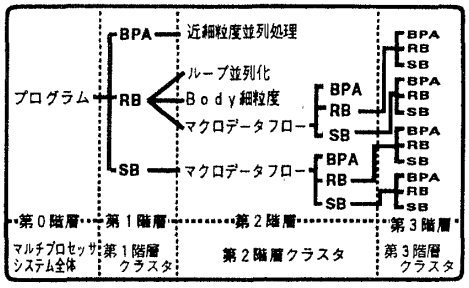

図1: マクロタスクの階層的な定義

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。本データローカライゼーションは、各階層のマクロタスクグラフ上で内部にサブMTを含まないMTに対して適用する。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てデータ授受を行えるように、データ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

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]を行う。その際、それらの部分ループのうち、各々が異なるPCに割り当てられるとPC間に多量のデータ転送が生じてしまう部分ループの集合は、実行時に同一PCにダイナミックスケジューリングすることによって、ローカルメモリを介したデータ授受を可能にする。以下に各手順を述べる。

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グループ内で、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。
4. TLG内で整合分割された部分RB集合の中で、データ転送量の多い部分RB集合を、データローカライゼーショングループとする。

例えば、図2(a)のTLGを、3PC用にループ整合分割法で分割すると、分割後のMTGは図2(b)のようになる。また、図2(b)において、各網掛け部分をデータローカライゼーショングループとする。

*A Data-Localization Scheme for Hierarchical Macro-Dataflow Computation

†Kenichi KOSHIZUKA, Akimasa YOSHIDA, Masami OKAMOTO, Hironori KASAHARA

†Waseda University. 3-4-1 Ohkubo Shinjuku-Ku, Tokyo 169

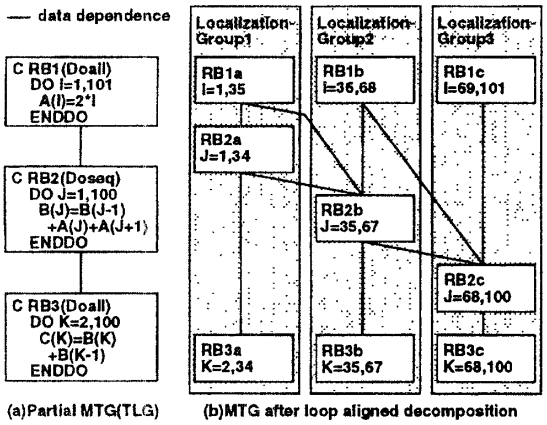

図 2: ループ整合分割の例

3.2 データ転送の解析

階層型マクロデータフロー処理は、PC に割り当てられた MT に対して PC 内の PE を使用して階層的に粗・中・近細粒度並列処理を実行することができる。MT がシーケンシャルループである場合、近細粒度並列処理を行うことによって、シーケンシャルループのイタレーション内およびイタレーション間で PC 内 PE 間データ転送が発生する可能性がある。また、データローカライゼーション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をアクセスするようにコード生成されるため、その配列変数の参照が発生するまでの間に、必要な範囲のデータがローカルメモリに配置されていなければならない。このため、当該プロセッサで参照前に定義されない範囲のデータは集中共有メモリからローカルメモリに転送する必要がある。以上のようなデータ転送の解析を行い、必要な転送命令を生成する。

3.3 データ転送時間の予測とコード生成

MT 内および MT 間の全ての配列変数に対してデータローカライゼーションの適用を行うと、集中共有メモリにデータを配置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、トータルの実行時間が延びてしまうことがある。そこで、本手法では、各配列変数ごとにデータローカライゼーションを適用する場合としない場合の推定データ転送時間を求め、そこから求められる推定トータル実行時間が短縮される配列変数だけにデータローカライゼーションを適用する。

データローカライゼーションを適用することが決定した配列変数に対しては、集中共有メモリ・ローカルメモリ間のデータ転送コードと、ローカルメモリへアクセスするコードを生成し、適用しない配列変数に対しては、集中共有メモリへアクセスするコードを生成する。

3.4 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

3.1 で生成されたデータローカライゼーショングループ内の複数の MT 間でデータローカライゼーションを実現するためには、これらの MT が実行時に同一の PC にダイナミックスケジューリングされなければならない。そこで、本手法では、Dynamic-CP 法を拡張し、コンパイラが指定した特定の MT 集合を同一の PC にスケジューリングする手法を提案する。本手法のダイナミックスケジューリングでは、コンパイル時に、あるデータローカライゼーショングループに属するいずれかの MT が PC に割り当てたられた時点で、そのデータローカライゼーショングループに属するその他の全ての MT に対して、必ず最初に割り当てられた MT と同一の PC に割り当てを行うようなルーチンを生成する。なお、最初に割り当てられる MT は、その時点で各 PC の推定負荷 (その PC で実行中の MT および割り当ての確定している MT の推定処理時間の総和) が最小の PC に割り当てられる。

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。OSCAR 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。

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) のプログラムを用いる。このプログラムを 1 プロセッサでシーケンシャル実行すると処理時間は 398[ms] であった。次に、4 プロセッサを使用して階層型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すると処理時間は 128[ms] となり、シーケンシャル処理と比べて 3.11 倍の速度向上が得られた。さらに、提案手法であるデータローカライゼーションを適用することで処理時間は 114[ms] となり、シーケンシャル処理と比べて 3.50 倍、階層型マクロデータフロー処理のみを適用した場合と比べて 12.9%の速度向上が得られた。

5 まとめ

本稿では、階層型マクロデータフロー処理において、それらの間で多量のデータ転送を生じる可能性がある複数の MT をデータローカライゼーショングループとして同一 PC に割り当て、それらの MT 間のデータ転送をローカルメモリを介して行う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 法を用いた性能評価の結果より、データローカライゼーションを適用しない階層型マクロデータフロー処理に比べて 12.9%の速度向上が得られ、提案手法の有効性が確かめられた。

今後の課題としては、階層型マクロデータフロー処理をスタティックスケジューリングと混合で実行する場合に、より広範囲かつ効率の良い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、文部省科学研究費補助金 (一般研究 (c)07680372) 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 Parallelizing 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] 笠原, 本多, 橋本: "OSCAR のアーキテクチャ", 信学論 D, Vol.38,No1, (1989-1).