1L－2

データ転送と処理のオーバラップを用いたデータ転送最小化自動並列化コンパイラ*

藤本謙作, 橋本茂, 笠原博徳†
早稲田大学理工学部電気工学科‡

1 はじめに

ローカルメモリ及び分散共有メモリを持つマルチプロセッサシステム上で、科学技術計算の並列処理を行なう際には、プロセッサ間データ転送オーバヘッドを最小化することが重要である。筆者等は、従来の逐次的プログラムを自動的にマクロタスクに分割し、データ転送オーバヘッドを最小化するようタスク実行及びデータ転送をスケジューリングした上で、除去できなかったデータ転送をデータポストストア技術を用いてプログラム実行とオーバラップさせ、そのオーバヘッドを最小化する手法を提案している[5]。本稿では、本手法を用いた並列化コンパイラの実マルチプロセッサスーパーコンピュータ上への実装について述べる。

2 データプレロード・ポストストアを用いたマルチプロセッサスケジューリング

本節では対象とするマルチプロセッサシステムのモデルと、データプレロード・ポストストアを用いたマルチプロセッサスケジューリングアルゴリズムについて述べる。

2.1 対象マルチプロセッサシステムのモデル

本稿では対象マルチプロセッサシステムとして、相互結合網(ネットワーク)を介して接続されたプロセッサエレメント(PE)が、それぞれローカルメモリ(LM)・分散共有メモリ(DSM)とデータ転送ユニット(DTU)を持つ分散共有メモリ型マルチプロセッサシステムを考える。ここでDTUとは他のPEのDSM上のデータへのリード・ライトを行なうデータ転送コントローラである。DTUによるデータの転送は、CPUのプログラム実行とオーバラップして行なうことができるものとする。

2.2 スケジューリング問題の定義

図1は、タスク(マクロデータフロー処理[1]のマクロタスクに相当する)間の先行制約を表したマクロタスクグラフ[1]を示している。各ノードがタスクを表し、ノード内の数字はタスク番号、左側の数字は実行時間、タスク間のエッジはデータ依存を表している。各エッジにはデータ番号とデータ転送量(転送に要する時間に換算したもの)が付加されている。但し、データ番号は複数の変数および配列中の領域の集合を指しており、対応するデータ転送量もそれらの転送に要する時間の総和である。

あるPE P_iが、あるタスク T_kを実行中に、P_i上で将来実行される他のタスク T_lが必要とするデータ D_j を、DTUを使用して D_jを定義したPE P_jのDSMから、P_iのローカルメモリへ転送することをプレロード[5]と言う。また、あるPE P_jがタスク T_jを実行してデータ D_jを定義し、この D_jを他のPE P_i上のタスク T_lが将来必要とする場合、PE P_jがタスク T_jの実行終了直後だけではなく、他のタスク T_kを実行中に、DTUを使用してPE P_jのローカルメモリからPE P_iのDSMにデータ D_jをストアすることを、ポストストアと言う。ポストストアはタスク T_lの実行が開始される時点までにストアを完了するようスケジューリングされる。データプレロード・ポストストアは、プログラム実行とデータ転送のオーバラップ(同時実行)を実現し、全体としてデータ転送オーバヘッドを最小化するために使用される。

本稿で取り扱うスケジューリング問題は、上述のような先行制約をもつ n 個のタスクを、m 台のプロセッサ上で処理する際に、プレロード・ポストストアも考慮して、全体の処理時間を最小にするスケジュール(割り当て、実行順序)を求める問題である。

2.3 スケジューリングアルゴリズム

データ転送時間を考慮するヒューリスティックアルゴリズムとして、本稿ではCP/DT/MISF(Critical Path/Data Transfer/Most Immediate Successors First)法[3]を用いる。CP/DT/MISF法は次のようなプライオリティ決定法に従うリストスケジューリングの一種で、タスクのPEへの割り当て時に、まずレディ(実行可能)タスク集合からCP(Critical Path[4])長の最も大きいタスクを選び出し、それらのタスクをデータプレロード・ポストストアを考慮して局所的なデータ転送時間が最小になる組合せでPEを割り当てる。CP長もデータ転送時間も等しいタスクとPEの組合せが複数ある場合は、直接後続タスク数の多いものを優先する。

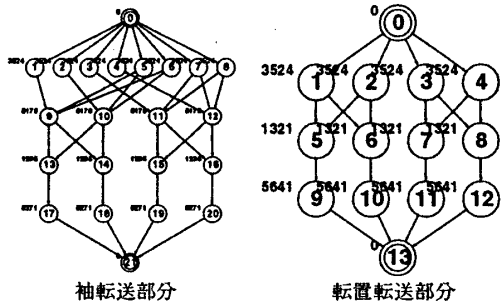

図1: 2分割された袖転送部分のタスクグラフ

このスケジューリングの際、以下のような方法でデータ転送を挿入する。たとえばあるスケジューリング時点 t にタスク T_iのPE P_iへの割当が決定されたとする。マクロタスクグラフのエッジ情報から、タスク T_iが必要とするデータ D_jと、それを定義するタスク T_jが分かる。スケジューリング時点 t までに生成されたスケジュールより、T_jの割り当てられたPE P_jがPE P_iと異なる時、タスク T_jの終了時点からスケジュー

*Data Transfer Optimizing Parallelizing Compiler Using Overlap of Data Transfer and Task Processing

†FUJIMOTO Kensaku, HASHIMOTO Shigeru, KASAHARA Hironori

‡Department of Electrical Engineering, School of Science and Engineering, Waseda University

リング時点 t までの間に、PE P_j から PE P_i へ DTU によるデータ転送 (プレロードまたはポストストア) が可能な期間があれば、この部分にデータ D_j の転送を挿入する。挿入できる期間がスケジュール時点 t までの期間にみつからなければ、タスク T_i の実行開始を遅らせて、データ転送を挿入する。あるスケジューリング時点で、ある二つのプロセッサ間でデータ転送が可能かどうかの決定は、PE 間の相互結合網のモデルに依存する。

3 通信と処理のオーバラップを考慮した並列化コンパイラの実装と性能評価

本節ではプレロード・ポストストアを考慮したスケジューリングアルゴリズムを利用した FORTRAN リストラクチャリングプリプロセッサを実装したマルチプロセッサシステムとその上で行なった性能評価について述べる。

3.1 実装対象マルチプロセッサシステム

今回対象とした実マルチプロセッサシステムは、富士通の仮想分散共有メモリ型マルチプロセッサスーパーコンピュータ VPP500 である[7]。VPP500 は最大で 222PE 構成が可能な MIMD マルチプロセッサシステムである。各 PE は最大 256M バイトの SRAM メモリ、LIW RISC プロセッサであるスカラユニットおよびベクトルユニットを持ち、最高処理速度は 1.6GFLOPS である。PE 間はクロスバネットワークで結合され、各 PE の持つデータ転送ユニット (DTU) が PE 間通信を処理する。DTU は単方向 400M バイト毎秒の PE 間データ転送を行なうことができる。各 DTU は PE のスカラユニットおよびベクトルユニットとは非同期的に動作する。このため、データ転送と計算処理は並列に実行される。

3.2 VPP500 用の自動並列化コンパイラ

本性能評価では、FORTRAN プログラムから VPP500 のバイナリを直接出力するコンパイラではなく、並列処理用のコンパイラディレクティブを持つ VPP Fortran ソースプログラムを出力するプリプロセッサを作成した。ただし以下の評価結果は、マクロタスクグラフより求めたスケジュール結果を効率良く実行するために、VPP Fortran の言語仕様外の動作に依存した同期指定記述を使用した。すなわち、VPP Fortran の言語仕様ではポストストア時に受信側 PE がデータ転送終了を待つ記述ができないので、非同期データ転送時に転送終了フラグを同時に転送する方式をとっている。

3.3 VPP500 上での評価

実装した自動並列化コンパイラで逐次的に書かれたプログラムを自動並列化し、実行時間の計測を行なった。評価に使用したプログラムは、航空流体解析に使われているもので、複数の DOALL 処理可能なループからなっており、x, y, z 方向に対応する 3 重ループを図 2 のように z 方向に PE 数で分割し、各 PE で並列処理した後、y 方向で分割して並列処理することができる。ただし、分割方向が変わると各 PE のローカルメモリの内容を入れ換えることになり、大量のデータ転送が生じる。これを転置転送と呼ぶ。また、各 PE の担当する分割されたデータ領域の境界付近で、隣接する PE 間でデータの交換が必要な場合がある。これを袖転送と呼ぶ。

本評価に当たってはプログラムの特徴である袖転送と転置転送の部分を使用した。

図 1 に、そのタスクグラフを示す。今回作成した自動並列化コンパイラ (プリプロセッサ) を用い、このプログラムを 4PE 構成の VPP500 上で実行して実行時間計測を行なった。

このとき本自動並列化コンパイラにより、自動的に対象プログラムのデータ転送およびタスク実行のスケジューリングを行ない、データ転送と処理をオーバラップさせてデータ転送最小化を施した。また一方、ユーザが VPP Fortran の並列化ディレクティブを挿入し、データ分割および並列化を指定したプログラムも作成して、実行時間の比較を行なった。後者においてはユーザが直接プログラムをチューニングしない限り、データ転送と処理のオーバラップによるデータ転送の自動最小化は行なわれないので、4 PE を使用して並列実行した場合、本自動並列化コンパイラを用いてデータ転送と処理をオーバラップさせ、データ転送最小化を行なった方が、袖転送の場合約 39%, 転置転送の場合約 32%程度プログラム実行時間が短縮された。転置転送では袖転送と比べてデータ転送が大量に発生するため、すべてのデータ転送をタスク実行とオーバラップさせることはできなかった。一方袖転送ではほとんどの転送をタスク実行とオーバラップさせることができたため、よりよい速度向上がみられた。

4 むすび

本稿では、VPP500 上にデータ転送と処理のオーバラップを用いてデータ転送の最小化を行なう自動並列化コンパイラを実装し、この手法の有効性を航空流体解析プログラムの特徴となる 2 種の部分プログラムを用いて検証した。現在は自動並列化コンパイラが、対象プログラム実行時の記憶容量最小化を行なっていないので、航空流体解析プログラム全体を動作させることはできなかった。今後大規模アプリケーションプログラムに適用し、性能を評価していく予定である。

最後に、VPP500 を使用させて下さった富士通株式会社に感謝致します。

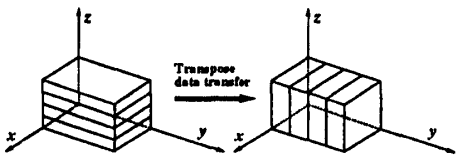

図 2: 航空流体プログラムにおける転置転送

参考文献

[1] Kasahara H., Honda H. and Narita S.: "A Multi-grain Parallelizing Compilation Scheme for OSCAR", Proc. 4th Workshop on Lang. and Compilers for Parallel Computing (Aug. 1991).

[2] Kasahara H., Honda H. and Narita S.: "Parallel Processing of Near Fine Grain Tasks Using Static Scheduling on OSCAR", Proc. IEEE Supercomputing '90(Nov. 1990).

[3] Kasahara H. and Narita S.: "Practical Multiprocessor Scheduling Algorithm for Efficient Parallel Processing", IEEE Trans. Comput., C-33, 11, pp. 1023-1029(Nov. 1983)

[4] Coffman E. G.: "Computer and Job-shop scheduling Theory", John Wiley & Sons (1976).

[5] 藤原 和典, 白鳥 健介, 鈴木 真, 笠原 博徳: "データプレロードおよびポストストアを考慮したマルチプロセッサスケジューリングアルゴリズム", 電子情報通信学会論文誌 (D-I), J75-D-I, 8 pp.495-503 (1992-8).

[6] 富士通株式会社: "VPP FORTRAN77 EX/VPP 使用手引書", (1994)

[7] 高村守幸: "富士通スーパーコンピュータの展開", 平成 7 年電気学会電子・情報・システム部門大会, A-10-1, 1995.8.

[8] Yoshida A., Maeda S., Fujimoto K. and Kasahara H.: "A Data-Localization Scheme using Task-Fusion for Macro-Dataflow Computation", IEEE Pacific Rim 1995 Conference, 1995.5.