4U－2 ワークステーション・クラスタ環境で粗粒度・細粒度並列処理を実現する並列化コンパイラ*

朝倉 宏一†　三田 勝史†　渡邊 豊英†
{asakura, sanda, watanabe}@watanabe.nuie.nagoya-u.ac.jp
名古屋大学大学院 工学研究科 情報工学専攻‡

1 はじめに

我々はワークステーション・クラスタ環境における並列処理のための自動並列化コンパイラを開発している．ワークステーション・クラスタ環境は，通常の並列計算機環境と比較するとネットワークの速度が遅いので，効率よい並列処理を達成するためには，適切な粒度のプロセスを生成しなければならない．したがって，ネットワークを介した通信頻度が低くなるようにプロセスを生成することが大きな目標になる．また，ワークステーション上でのプロセス起動オーバヘッドも大きいので，それを考慮したプロセス粒度を採用しなければならない．

我々は，プログラム中の関数をプロセス生成の単位としたプログラム並列化手法を提案している [1]．関数がプロセス生成単位となるので，各プロセスの変数空間は互いに独立となる．したがって，ネットワークを介したプロセス間通信の頻度を抑えることができる．また，プロセス起動オーバヘッドに対処するため，複数の関数をまとめて一つのプロセスとする手法を採用した．この並列化手法により，ワークステーション・クラスタ環境において効率よく実行可能なプロセスを生成することができる．

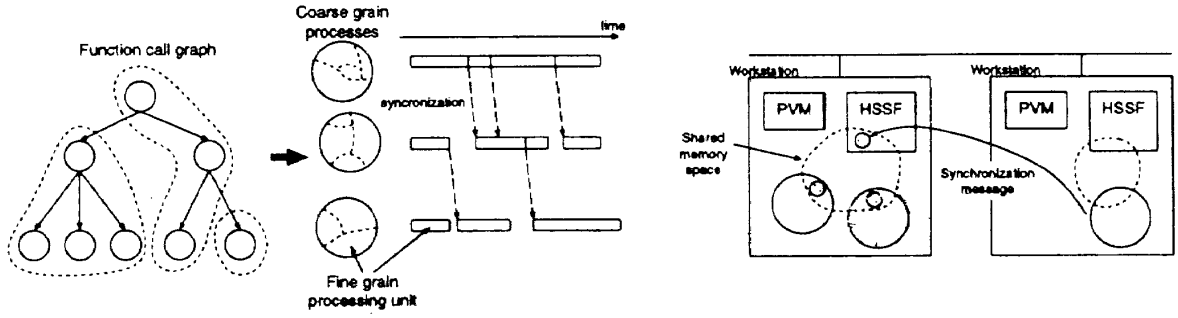
図1(a) にプログラム並列化手法の概念図を示す．図1(a) は，プログラムの関数手続き間の呼出し関係を表した関数呼出しグラフである．関数手続き間の呼出し関係における並列実行性を抽出し，プロセスを生成する．

*Coarse-grain / Fine-grain Parallel Processing Method on Workstation Cluster Environment
†Koichi ASAKURA, Katsushi SANDA and Toyohide WATANABE
‡Department of Information Engineering, Graduate School of Engineering, Nagoya University

2 細粒度並列性抽出のためのプロセス間同期制御

上記のプログラム並列化手法により生成されたプロセスは，ワークステーション・クラスタ環境で効率よく実行できる．しかし，上記の手法ではプログラムの並列実行性が十分に抽出されないという問題がある．すなわち，関数手続き単位でプロセスを生成するので，関数内の基本ブロックや文間の細粒度並列性は抽出されず，プログラムの並列実行性が制限されるという問題が発生する．これは，粗粒度プロセス間における同期処理が並列実行性を阻害していると見ることができる．つまり，プロセスの粒度が粗いとき，プロセス間のデータ送受信によるプロセス間同期によって，プロセスの実行中断状態が生じる．これにより，プロセスが並列実行されず，効率のよい並列処理を達成することができない．

プロセス間同期によるプロセス実行中断状態を回避し，細粒度並列性を最大限活用するために，我々はプロセス間同期をプロセス単位ではなく，プロセス内の文単位で制御することを考える．一般にプロセスが起動可能となるには，そのプロセスに関する制御依存関係，データ依存関係がすべて満たされなければならない．つまり，そのプロセスが実行されることが確定し (制御依存関係)，そのプロセスで使用されるデータがすべて参照可能 (データ依存関係) でなければならない．ここで，データ依存関係に着目すると，プロセス内の各文の実行には必ずしもすべてのデータ依存関係が満足される必要はない．プロセス内には一部のデータ依存関係が満足されていれば実行可能な文や，データ依存関係が満足されていなくても実行可能な文が存在する．したがって，参照するデー



(a) coarse grain process generation

(b) extraction of fine grain parallelism

(c) Inter-process synchronization mechanism

図 1: プログラム並列化，並列プロセス実行の概念図

タの準備状況に従って，これらの文の実行を制御することで，プロセスの実行中断状態を最小限にし，粗粒度プロセスの効率よい並列実行が可能となる．

我々はこのプロセス実行制御を漸進的並列処理と呼ぶ [2]．漸進的並列処理により，データの準備状況に従って粗粒度プロセスを部分的に実行可能となり，プロセス実行中断状態を回避することができる．すなわち，プロセス間の細粒度な並列実行性が抽出され，効率よい並列処理を達成できる．

図 1(b) に漸進的並列処理の概念図を示す．粗粒度プロセス内には，プロセス間のデータ依存関係の解析結果により，細粒度並列実行単位が生成される．この細粒度並列実行単位がプロセスの実行によるデータの準備状況に従って漸進的に実行され，プロセスの実行中断状態を最小限にする．

3 プロセス間高速同期機構

粗粒度プロセスに対して漸進的並列処理を適用することにより，プロセス間で発生する同期処理の頻度が高くなる．すなわち，プロセス間同期のために授受される同期メッセージの量が増大する．したがって，上記の漸進的並列処理を効率よく行うためには，同期メッセージの増加により処理効率が低下しないような機構が必要となる．我々は，ワークステーション・クラスタに PVM を使用した並列処理環境を想定しているが，同期メッセージ処理に PVM のメッセージ通信を使用すると，通信オーバヘッドが大きくなり，漸進的並列処理が達成できない．そこで我々は高速なプロセス間同期を達成するために，PVM とは独立なプロセス間高速同期機構 (HSSF) を提供する．図 1(c) に HSSF の仕組みを示す．HSSF は他プロセスからの同期メッセージを受信する役割を持つ．これは粗粒度プロセスにおける同期メッセージ受信処理のオーバヘッドを削減するためである．また，HSSF と粗粒度プロセスは OS の提供する共有メモリ機構を利用し同一変数空間を共有している．そして，同期メッセージの到着状況は，この共有された変数空間を参照することにより確認することができる．共有メモリ機構を利用することにより，HSSF と粗粒度プロセス間における同期メッセージ送受信処理を削除し，通信オーバヘッドを削減することができる．

4 おわりに

本論文では，ワークステーション・クラスタ環境において効率のよい並列処理を達成するための，プログラム並列化手法，プロセス間同期制御手法，および高速プロセス間同期機構について述べた．我々は現在，この考えに基づいた自動並列化コンパイラと，PVM に基づいた並列処理環境を実装中である．

参考文献

[1] 朝倉宏一, 渡邉豊英: “ワークステーション・クラスタ環境における並列化コンパイラの構成”, 電気学会論文誌 C, Vol. 118-C, No. 4, pp. 558-568 (1998).

[2] 朝倉宏一, 渡邉豊英: “プロセス間同期オーバヘッド削減のための漸進的並列処理手法の提案”, 並列処理シンポジウム JSPP'98, p. 148 (1998).