並列オブジェクト指向トータルアーキテクチャA-NET
2L－6　－WSクラスタへのA-NETLの実装－

齋藤宣人　古田貴寛　月川淳　馬場敬信　吉永努
宇都宮大学工学部*

1. はじめに

近年，ネットワークで結合されたワークステーション（以後，WSクラスタ）を統合的に扱い，より高性能な並列分散処理を目指す研究が活発に行われている．また，このための標準的なメッセージパッシングライブラリとしてPVM3[1]，MPI，p4などの提案がある．

本研究では，並列オブジェクト指向概念のもとに統合された並列分散プログラミング環境をユーザに提供するために，我々が設計した並列オブジェクト指向言語A-NETL[2, 3]の処理系をWSクラスタ上に実現することを目的とする．

本稿では，処理系の設計方針，実現方法，初期評価，及び，最適化の構想などについて述べる．

2. 設計方針

処理系は，トランスレータ，アロケータ，ホストプログラムによって構成される．

(i) トランスレータ

トランスレータの実現に当っては，可搬性を重視し，CとPVM3を用いる．

(ii) アロケータ

アロケータは，メッセージ通信によるオーバーヘッドを最小限におさえることを目的とし，オブジェクト間関係，WSの性能などを考慮して，WSへのオブジェクトの割り付けを行う．

(iii) ホストプログラム

ホストプログラムはプログラムを構成するオブジェクトをPVMタスクとして起動し，ユーザから入力されたメッセージの解析・送信を行うユーザインタフェイスである．また，A-NET計算機におけるHOSTとして特化された機能を実現する．

これらの処理系は，A-NETが提供するAPSS(A-NETL Programming Support System)上で統括して扱われる．

3. 実現方法

今回，PVM3を用いてWSクラスタ上にA-NETLの処理系を実現するにあたり，特に以下の3点を考慮した．

(i) 変数の型の取り扱い

A-NETLでは，変数の型をユーザが陽に指定する必要がなく，自由に扱える．従って，PVM3を利用したCプログラムにトランスレートするには，セマンティクスの整合をとるための工夫が必要である．そこで，A-NETLと同等の動作を提供するために，全ての変数を構造体を用いて実現する．

(ii) コンテクストの実現

プログラムの実行イメージは全てコンテクストと呼ばれるデータ構造の形で保持する．このコンテクストには，一時変数，引数配列，オブジェクトIDなどの情報が含まれ，コンテクストチェンジによりオブジェクト内並列を実現している．また、サスペンドの掛かる可能性のある場所にラベルを挿入し，そのラベルによってプログラムの実行再開のアドレスを保持している．

(iii) メッセージの実現

A-NETLの提供するメッセージの形態には，非同期型の過去型と，同期型の現在型・未来型の3種類がある．このうち，過去型メッセージはPVM3が提供する非同期ブロッキングメッセージ送信関数pvm_send()によって，容易に実現できる．他の同期をとるメッセージでは，返値の確認をするためのサスペンド機構が必要である．これは，サスペンドされる可能性がある全ての変数に対して，返値が設定されたか否かを確認する関数を呼ぶことで実現している．サスペンドによって生成されるコンテクストはレディリストに積まれ，実行可能になるまで待機する．

4. 初期評価

実装を行う前に，PVM3によるメッセージ通信時間の測定と前節(i)のための実験を行った．

(i) メッセージ通信時間の測定

PVM3によるメッセージ通信のオーバーヘッドがどの程度あるかを調べるために，2つのホスト間で整数を送り合うプログラムを作成し，PVM3を用いたものとソケット通信を用いたものを用意した．平均実行時間は，PVM3を用いた場合250.36 $msec$であり，ソケット通信を用いた場合は31.71 $msec$であった．

この結果より，PVM3の通信オーバーヘッドがかなり大きいものであることが分かる．

(ii) 型推測による実行時間の変化

変数を全て構造体として実現する方法は，四則演算，メッセージ引数のパック／アンパック，メッセージ通信などからみて，かなり大きいオーバーヘッドが生じる．

*A parallel object-oriented total architecture A-NET – Implementation of a parallel object-oriented language for workstation clusters –,Norihito SAITOH,Takahiro FURUTA,Atsushi TSUKIKAWA,Takanobu BABA,and Tsutomu YOSHINAGA,Utsunomiya University.

そこで，変数の型を推測することにより，実行時間がどの程度削減されるかを調べた．

A-NETL で書かれた N Queen，Life Game，Road Map の 3 種類のサンプルプログラムを用意し，ハンドトランスレートした．作業の効率を上げるために，C++ を用いライブラリを作成し利用した．これらのプログラムについて，変数への値の代入により型を推測するルールを用意し，各プログラムについて

(a) 全ての変数を構造体を用いて定義したもの

(b) 型推測ルールに従って型推測を行い，推測できない変数を (a) 同様に定義したもの

(c) 全ての変数について最適に型の宣言をしたもの

それぞれ 3 種類を作成した．

これらのプログラムを，単一ホスト上で 100 回実行したときの平均実行時間をまとめたものが，表 1である．また，複数ホスト上で処理を分散して実行したときの結果を表 2にまとめてある．

表 1: 単一ホストにおける平均実行時間

単一ホスト [sec]			
	N Queen	Life Game	Road Map
(a)	6.81	6.17	—
(b)	6.16	5.08	—
(c)	4.79	5.03	—

表 2: 複数ホスト (2 ホスト) における平均実行時間

複数ホスト (2 ホスト)[sec]			
	N Queen	Life Game	Road Map
(a)	8.24	5.90	2.03
(b)	7.79	4.71	1.97
(c)	7.01	4.32	1.90

表 1において Road Map の実行時間が欠けているのは，Road Map のオブジェクト総数が 36 と多く，単一ホスト上で実行できなかったためである．

表 1の結果，N Queen が型推測によって 650$msec$ の速度向上が達成された．また，Life Game では，型推測によって最も実行時間が削減されており，1090$msec$ の速度向上がなされている．実験に用いた Life Game は，処理サイクルを 10 回実行しているので，1 サイクルあたり実行時間が 18%程度削減されたことになる．

表 2の結果でも，型推測による速度向上が達成されている．N Queen の実行結果には台数効果が現れていない．これは，N Queen の全てのオブジェクトが，相互にメッセージの通信を行っているため，処理分散による効果よりもメッセージ通信の遅延が増大したためである．Life Game は，全体の約 18%のメッセージ通信に遅延が生じるものの，処理の分散による効果がそれを上回っているため，台数効果が現れている．Road Map の処理速度の向上があまりなされていない原因は，メッセージ通信の割合が高いことによる．

これらのサンプルプログラムの規模が小さいことを考えれば，より大規模なプログラムに於いては更なる速度向上が期待できる．

5. 最適化の構想

以上の実験結果をふまえ，最適化の構想として以下の 2 点を挙げる．

(i) 通信オーバーヘッドの削減

メッセージ通信の性能を上げることが，処理速度の向上のために重要である．

このためには，PVM3 を始めとするメッセージパッシングライブラリそのものの高速化が前提となる．

(ii) 型推測

変数の型推測は，上記の実験結果より規模の大きいプログラムでその効果が大きく得られることが予想される．

6. おわりに

並列オブジェクト指向言語 A-NETL を WS クラスタに実現するための，トランスレータの設計方針を述べ，最適化の構想を示した．

現在，トランスレータは C のコードにトランスレートするところまで完成し，ホストプログラム及びアロケータの完成を待つ状態にある．

今後は，MPI を始めとする他のメッセージパッシングライブラリへの対応と，トランスレータの最適化を行い，より使いやすい並列分散処理環境を提供したいと考えている．

謝辞

本研究は，一部文部省科学研究費，一般(C)07680334，奨励 (A)07780225，および電気通信普及財団の援助による．

参考文献

[1] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderam,"PVM3 USER'S GUIDE AND REFERENCE MANUAL", ORNL/TM-12187, Sep 1994.

[2] T. Baba, T. Yoshinaga, T. Furuta,"Programming and Debugging for Massive Parallelism: Case for a ParallelObject-Oriented Language A-NETL ",Proc. Workshop on Object-Based Parallel and Distributed Computation (OBPDC'95),1995.

[3] T. Baba, T. Yoshinaga, "A-NETL: A Language for Massively Parallel Object-Oriented Computing",Proc. Working Conf. on Massively Parallel Programming Models (MPPM'95),1995.