並列オブジェクト指向トータルアーキテクチャ 1L-5 A-NETのプログラムデバッグ方式

濱田 正泰* 南斉 清巳** 吉永 努* 馬場 敬信*
(*宇都宮大学工学部 **小山高専)

1. はじめに

並列処理のプログラムは、その設計，テスト，および管理において、逐次処理の場合とは異なった概念や手法を必要とする。従って、プログラムの生産性，信頼性を確保し、並列処理システムを効率よく稼動させるためには、並列プログラムの動作の観測やプログラムの評価などの機能を備えた支援環境[1]が必要となる。

我々は、並列処理システムとして、並列オブジェクト指向トータルアーキテクチャA-NET[2]を設計開発中であり、そのプログラム記述言語として、並列オブジェクト指向言語A-NETL[3]を定義した。本稿ではA-NETLにより記述されたプログラムのデバッグ方式について述べる。

2. A-NETLデバッガの設計方針

我々は、A-NETLにより記述されたプログラムのデバッギングを支援する、A-NETLデバッガの設計／試作を進めている。並列処理システムにおけるデバッギングに固有な問題を明確にし、その対策を検討することにより、A-NETLデバッガの設計方針を次のように決定した。

（1）メッセージの履歴による動作の再現

一般に複数の処理要素が非同期に異なった処理を行っている場合、動作過程を再現することは非常に困難である。

この問題に対し、A-NETがオブジェクト指向であることから、その遷移過程はメッセージパッシングにより決定されることを利用する。すなわち、メッセージの履歴をたどることにより遷移過程の半順序関係を決定し、その関係内での動作の再現を可能とする。

（2）分散デバッガモデル[4]を用いた負荷分散

並列システムを構成するシステムサイズは大規模になる傾向にある。このようなシステム、特に高並列システムを一元的に管理することは、システムの処理能力のボトルネックになる。また、処理の遷移が通信により行われるような並列システムでは、通信コストの増大は並列処理の長所を相殺してしまうことになる。

これらの問題に対し、図1に示すような分散デバッガモデルを採用し、デバッグ操作の負荷分散を行い、ローカルデバッガに可能な限りの機能を与え、なるべくマスタデバッガとの通信を行わずに、デバッグ処理が行えるようにすることにより、負荷の集中，処理能力の低下，および通信コストの増加を回避する。

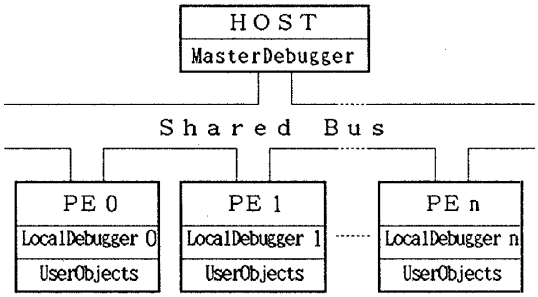

図1. A-NETLデバッガの配置イメージ

（3）高度なユーザインタフェースの構築

高並列システムにおけるデバッグ操作では、その操作に必要とされる情報量が爆発的に増加することが予想される。

この問題に対し、ユーザに提供すべき情報に重みづけを行い、さらにその提供にあたっては、図形，色彩，およびアニメーションを用いて情報を表現することにより、高並列システムの動作状況の理解を支援する。

3. A-NETLデバッガの基本設計とその実現

前節で示した設計方針をもとに、A-NETLデバッガの基本設計を次のように行った。

・機能分割

分散デバッガモデルの採用に伴い、ローカルデバッガとマスタデバッガの機能分割を行なう。

（1）ローカルデバッガ

次のような機能をもたせる。

①デバッグモード／通常モードの切替え機能

ここで、デバッグモードとは、プログラムの実行時に、ファームウェアレベルでデバッグ操作を行う状態を指す。デバッグモードフラグをファームウェア内に設け、このフラグを操作することにより、デバッグモードと通常実

Program debugging scheme for a parallel object-oriented total architecture A-NET
Masahiro HAMADA*, Kiyomi NANSAI**, Tsutomu YOSHINAGA*, Takanobu BABA*
*Utsunomiya University. **Oyama National College of Technology.

行モードの切替えを行う。

②イベントの設定／検索機能

ＯＳレベルデバッガを起動する事象をイベントと呼ぶ。必要となるイベントの種類は処理システムやユーザの要求に大きく依存する。従って、A-ＮＥＴＬデバッガのイベント仕様は固定とせず、ファームウェアレベルでその仕様を容易に変更できるように、プログラマブルな構造をもつデータとして定義する。現在イベントの種類として次のものを考えている。

・ソースプログラム上で指定した行の実行
（ブレークポイント設定機能）
・メソッド単位の実行　（トレース機能）
・変数のアクセス　（変数のウォッチ機能）

これらのイベントは各ＰＥのローカルメモリ上にリスト形式で設定される。

③メッセージ送受信履歴の保存機能

並列性に起因する問題のデバッグを行うために用いる。プログラムの実行過程を再現するため、履歴をとるべきメッセージにはタイムスタンプが付加される。

ローカルデバッガは、次の２つで構成される。

（1.1）ファームウェアレベルデバッガ

イベントリストから、ＰＥの状態と一致するものを検索する。一致すれば内部割り込みを発生し、ＯＳレベルデバッガへ制御を移す。

（1.2）ＯＳレベルデバッガ

イベントトラップによって起動される、ローカルＯＳ[6]内の機能である。起動されるとイベントが発生したことをホストに伝え、ユーザプログラムの実行を中断する。そのほか、ローカルメモリへのアクセス機能，実行モードの切替え機能，イベントリストの変更機能，メッセージ受信履歴の保存機能などがある。

図２は、ユーザオブジェクトのデバッグモードでの状態遷移を表している。ユーザオブジェクトの実行時に、ＰＥの状態がイベントリストに記述されている状態と一致したとき、ユーザオブジェクトの実行は中断され、ローカルデバッガからマスタデバッガへの通信が行われ、デバッグ処理が行われる。

（２）マスタデバッガ

マスタデバッガではローカルデバッガの機能を組み合わせて、主に並列性に依存する問題への対応、ユーザインタフェースに関する部分を実現する。具体的には次の機能を実現する。

①メッセージ履歴を用いたデバッグ機能

ローカルデバッガのメッセージ送受信履歴の保存機能を利用して、プログラムの状態遷移を提供し、その状態を再現する。

②ソースコードレベルでのイベント設定機能

ソースコード上からイベントを設定することを可能とする。

③図形表示などによる表示機能

例えば、動的なオブジェクトの生成／消去，メッセージパッシングの様子などの過程を、アニメーションなどの技術を用いて表示する。

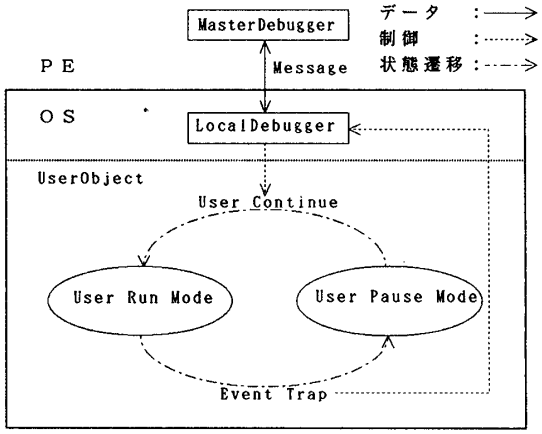

図２．ユーザオブジェクトの状態遷移

４．おわりに

本稿では、A-ＮＥＴにおけるプログラミング支援環境の一部として、A-ＮＥＴＬデバッガの設計／実現法について述べた。

本デバッガは、分散デバッガモデル、プログラマブルイベントデータ構造を採用したことにより、負荷分散によるデバッグ作業の効率化や、将来的な要求に柔軟な対応が可能になっている。ハードウェアデバッガ（マイクロ診断等）はA-ＮＥＴＬデバッガとは別に設計中である。今後はマスタデバッガの実現と、A-ＮＥＴＬデバッガ全体の評価を行う予定である。

参考文献

[1]加賀谷 他:"並列オブジェクト指向言語A-NETLのプログラミング支援環境",情報処理学会第38回大会,4P-2(1989).
[2]馬場 他:"並列オブジェクト指向トータルアーキテクチャA-NET",並列処理シンポジウム'89,A4-1(1989).
[3]岩本 他:"並列オブジェクト指向言語A-NETLの言語処理系",情報処理学会第38回大会,4P-1(1989).
[4]H.G.Molina, F.Germano: "Debugging a Distributed Computing System",IEEE Trans. Software,Vol. SE-10, No.2,pp.210-219,(1984-3).
[5]吉永 他:"並列オブジェクト指向トータルアーキテクチャA-NETのローカルOS",情報処理学会第45回オペレーティングシステム研究会報告,89-OS-45-5(1989).