並列オブジェクト指向トータルアーキテクチャ A-NETのシミュレータ

1H−8

濱田 正泰　吉永 努　馬場 敬信
(宇都宮大学工学部)

1．はじめに

現在、我々の研究室では、並列オブジェクト指向概念に基づいたトータルアーキテクチャA-NETの研究を行なっている[1]。本研究は、分散環境にある UNIX ワークステーション上に、ネットワークワイドなA-NET計算機のシミュレータを構築し、A-NETの計算モデルやハードウェア構成などの評価を行うことを目的としている。

本稿では、A-NET計算機の概要について述べ、シミュレータを構築するためのモデル化を行う。また、分散メモリ型並列計算機のシミュレータを試作するための基礎技術や実現方法について述べる。最後に、A-NETを評価するための項目の選定を行い、今後の指針とする。

2．A-NET計算機の概要とモデル化

シミュレータを構築する上で注目すべきA-NET計算機の主な特徴を次に挙げる。

(1)MIMD型マルチコンピュータシステム

A-NET計算機は、並列オブジェクト指向言語A-NETLの高速実行を行うための 320KB の局所メモリを持つノードプロセッサ(以下単にノード)を、1000 台規模ネットワークにより結合した高並列計算機である。

独立した処理要素であるノードは、オブジェクトのメソッド実行を行うプロセッシングエレメント(PE)と、メッセージの経路選択やオブジェクトコードの転送を行うルータから構成され、ネットワークを介し協調して処理を行う。

(2)ネットワークトポロジ独立

広範な問題領域に適応できるように、そのノード構成はネットワークトポロジ独立なものである。このため、メッセージの経路選択は固定的なものとせず、トポロジに応じて変更可能なものとなっている。

(3)A-NETL指向高機能命令の実装

A-NET計算機の設計は、実行モデルやA-NETLからのトップダウンアプローチにより進められてきた。従って、その命令セットアーキテクチャはA-NETLを効率よく実行できるように定義されており、特にメッセージパッシングを重視した設計になっている。また、動的オブジェクト生成時にはオブジェクトコードのノード間転送が生じるため、その命令セットを高機能なものに設定し、この時の通信コストを抑えている。

以上のような特徴を持つA-NET計算機を UNIX ワークステーション上でシミュレートするために、次のようなモデル化を行った。

(1)マルチプロセス

並列計算機を自然にモデル化するため、並列動作を行う機能単位にプロセスを割り当てる。すなわち、フロントエンド、及び、各ルータとPEにプロセスを割り当てる。また、ネットワーク制御など、シミュレータを実現する上で必要となる機能単位にプロセスを割り当てる。

(2)プロセス間通信

A-NET計算機は3つの通信形態を持つ。その通信形態をプロセス間通信としてモデル化したものを次に示す。

①共有バス結合

フロントエンド-ルータ間の通信形態である。一本のバスに対しフロントエンド及び各ルータが接続されるため、バスの排他制御が必要となる。また、送信先を決定するため、データバスの他に制御線に相当する機構が必要になる。しかし、分散環境にある UNIX の各ホスト間で効率の良い同一のオブジェクト(UNIX におけるデータ入出力の対象となるもの)を持つことはむずかしいため、共有バスを実現するのは困難である。従って、A-NETの構成要素としての共有バスがA-NETに与える影響は少ないという理由から、フロントエンドと各ノードの結合は木構造を持った結合とする。

②リンク結合

ルータ-ルータ間の通信形態である。ルータは各隣接ノードとの通信用に、1対1に結合される専用のポートを持つ。シミュレータではこのポートをソケットにより実現する。

③共有メモリ結合

ルータ-PE間の通信形態である。本来は共有メモリを用いるべきであるが、シミュレータを実現するワークステーション上の UNIX(NEWS-OS 3.3)にこの機能がないため、ファイル上に共有メモリを実現し、ファイルロック機能を用いてその排他制御を行う。

Simulator for a parallel object-oriented total architecture A-NET
Masahiro HAMADA, Tsutomu YOSHINAGA, Takanobu BABA
Utsunomiya University.

3．シミュレータの構成

図1にシミュレータの構成を示す。

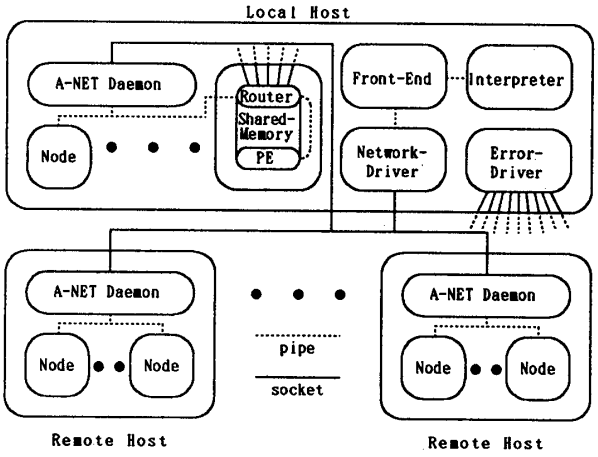

図1：プロセス構成とプロセス間通信

図1に基づいた UNIX ワークステーション上でのシミュレータの実現法を次に示す。

(1)プロセス構成

シミュレータは次に示すプロセス群により構成される。

フロントエンドプロセスはプログラムの初期ロード、ユーザインタフェースの実現などを行う。

ルータプロセスはネットワークの生成、メッセージのルーティングなどを行う。

ＰＥプロセスはメソッドの実行を行う。

そのほか、フロントエンド-ノード間結合を実現するネットワークドライバとA-ＮＥＴデーモン、エラーメッセージ等の表示を行うエラードライバ、ユーザインタフェースのためのA-ＮＥＴＬサブセットインタプリタにプロセスを与えている。

(2)プロセス間通信の実現

プロセス間通信を実現するための基礎技術として次のものを用いた。

①パイプ

フロントエンド-インタプリタ間、フロントエンド-ネットワークドライバ間、A-ＮＥＴデーモン-ルータ間、ルータ-ＰＥ間の通信チャネルとして使用している。

②ソケット

ネットワークドライバ-A-ＮＥＴデーモン間、ルータ-ルータ間通信チャネルとして使用している。これらの接続は、プロセス間に親子関係がなく任意の接続を要求されるため、パイプではなく tcp によるソケットを使用する[2]。これにより任意のプロセス間がネットワークを介して接続可能になる。また、各リモートホスト上のA-ＮＥＴデーモン、ルータ、ＰＥからのエラーメッセージは、これらが制御端末を持たないため、ネットワークドライバから起動されるエラードライバを介してシミュレータを起動した端末へ出力される。この場合のプロセス間通信として udp によるソケットを利用している[2]。

③ファイルロック

ルータ-ＰＥ間は割り込み等の機能制御用のパイプを用いた通信の他に、ファイルベースの共有メモリを用いることによりA-ＮＥＴの共有メモリを実現している。ルータとＰＥは独立したプロセスとなっているため、共有メモリへのアクセスは非同期的に行われる。従って、共有メモリにアクセスする際には排他制御を行う必要がある。共有メモリがファイルベースで実現されているため、UNIX のシステムコールである flock を用いて排他制御を実現する。

(3)ノードの割当

分散環境を利用した計算機環境で複数のタスクを動作させる場合、効率的な負荷分散を行う必要がある。A-ＮＥＴシミュレータにおける負荷分散は、各 UNIX ワークステーションへの割当ノード数を変化させることにより実現される。割当ノード数は、基本的に、各ホストの処理能力とシミュレータ起動時のロードアベレージにより決定される。各ホストの処理能力は、経験を基に静的な比としてデータベースに登録している。また、各ホストのロードアベレージは、UNIX のデーモンである rwhod が管理するデータベースを利用し動的に決定している。ただし、各ホストへのノード割当が決定した後のノード割当の変更は行っていない。

4．おわりに

今回試作したシミュレータは、ワークステーションの性能や UNIX の制限を考えると、扱えるホスト数が約60台、各ホスト上で扱えるノード数が約20ノードとなり、理想的には合計1200台程度のシミュレーションが可能となる。この値はA-ＮＥＴが目標とするネットワーク規模を満足する。

計算機のシミュレータを試作する主な目的は、計算モデルの妥当性や、計算機を構成する各ファシリティの性能バランスを求めることにある。今後は、ルータとＰＥの実行性能比によるシステム全体の実行性能への影響や、ネットワークトポロジの変化による実行性能への影響[3]、台数効果による実行性能の影響などを実際にシミュレータを用いて評価していく予定である。

参考文献

[1]Baba, et al.,"A Parallel Object-Oriented Total Architecture: A-NET,"Proc. Supercomputing '90, pp.278-285(1990).

[2]斉藤明紀,山口 英:"UNIX Communication Notes," UNIX MAGAZINE(ASCII)Apr.-Oct.1990

[3]Baba, et al.,"A Network-Topology Independent Task Allocation Strategy for Parallel Computers, "Proc. Supercomputing'90, pp.878-887(1990).