計算機アーキテクチャ　82－4
（１９９０．４．２０）

ランデブと共有変数を持つ並列型言語の実行支援系

平原正樹[†]　岡村耕二[††]　縄田毅史[††]　荒木啓二郎[††]

[†]九州大学　中央計数施設

[††]九州大学　情報工学科

〒812 福岡市東区箱崎6-10-1

メッセージ型通信機能とメモリ共有型通信機能を併せ持つAda言語サブセット処理系ParaDisE、およびParaDisEで記述された並列プログラムの実行を支援する分散オペレーティングシステム核DaOSの実現について述べる。DaOSは共有メモリの有無に関わらず、システム全体で単一のアドレス空間を提供し、ParaDisEはこの機能を利用して、ランデブおよび共有変数を実現する。数値計算に代表される定型的な並列計算を負荷分散で高速化すると共に、粗結合･密結合入り交じる環境上で一様な並列プログラミング環境を提供する。現在、ParaDisEおよびDaOSはイーサネットで接続されたUNIXワークステーション上で動作している。本稿では、ParaDisE/DaOSの概要と共に、作成した処理系上での基本操作の測定データも報告する。

ON SUPPORTING THE PARALLEL PROGRAMMING LANGUAGE WITH RENDEZVOUS AND SHARED VARIABLES

Masaki HIRABARU[†]　Koji OKAMURA[††]　Takeshi NAWATA[††]　Keijiro ARAKI[††]

[†]Computation Center, Kyushu University

[††]Department of Computer Science and Communication Engineering, Kyushu University

6-10-1 Hakozaki, Higashi-ku, Fukuoka 812, Japan

Email: hi@kyushu-u.ac.jp

We describe our implementation of the parallel programming language ParaDisE and the distributed operating system DaOS. DaOS supports uniform address spaces over the parallel/distributed systems without shared memories. ParaDisE is a subset of the Ada programming language and provides rendezvous and shared variables on the uniform address space. They help speed-up of parallel computations with balancing load. In this paper, we show our current implementation and report its efficiency on loosely-coupled UNIX workstations with a LAN.

1. まえがき

我々の研究の目的は、並列･分散アプリケーションの記述を通して、並列プログラミング言語の提供すべき機能を見直すことにある。現在の並列プログラミング言語は並列･分散アプリケーションを記述するのに十分か。あるいは、殆ど使われない機能の提供に固執していないか。実際のアプリケーションの多くは複雑な同期･通信機構よりも負荷分散のための機能の方が重要かも知れない。具体的な並列プログラミング言語を想定しない分散オペレーティングシステムの有用性の議論が不毛であると同様に、アプリケーションを考えない並列プログラミング言語の有効性の議論も説得力に欠ける。

我々は、粗結合･密結合両方の形態が混在するシステムを対象とし、単一の言語および処理系で、それらの上の並列･分散アプリケーションの記述を可能としたい。このことは、必ずしも言語仕様の肥大化を容認するものではない。並列プログラミング言語Ada[1]の持つ並列処理機能(タスク･ランデブ･共有変数)を提供するAda言語サブセットを設定し、その処理系を実現する。また、Ada言語が提供する並列処理機能に対しては、幾つかの欠点が指摘され、拡張が提案されている[2]。実際のアプリケーションの記述を通してこれらの欠点を確認し、拡張案について実現･考察を行いたい。

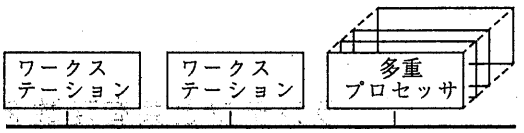
我々はまず、アプリケーションとして数値計算に代表される定形的な並列計算を想定する。コンパイラの静的解析情報をオペレーティングシステムに伝えることで、負荷分散を効率良く行いたい。その手始めとして、ローカルエリアネットワークで結合した分散システムで、Ada言語サブセット処理系ParaDisE[3,4]およびParaDisEで記述された並列プログラムの実行を支援する分散オペレーティングシステム核DaOS[3,5]を実現した。本稿では両システムの概要および評価について報告する。

2. 並列･分散プログラミングの問題点

対象とする並列･分散システムのモデルを示すと共に、そのシステム上での並列･分散プログラミングの問題点を指摘する。

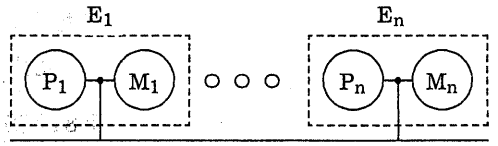
2.1 対象とする並列･分散システムのモデル

図1は我々が対象とする並列･分散システムのモデルである。多数のワークステーションと多重プロセッサシステムがローカルエリアネットワークで結ばれている。多重プロセッサにはハードウェアによる分散共有メモリが存在し、粗結合部分･密結合部分双方に局所参照と遠隔参照のコストの違いが存在する。前処理･後処理をワークステーション上で分散処理し、並列計算は主に多重プロセッサ上で共有メモリを活用して行うのが想定する典型的な利用法である。



ローカルエリアネットワーク

(a) 全体図



相互結合網

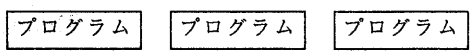
P_i: プロセッサ　M_i: メモリ　E_i: プロセッサ要素

(b) 多重プロセッサ部分(分散共有メモリ)

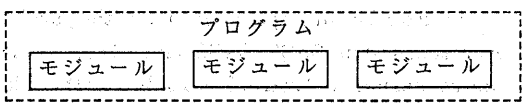
図1　対象とする並列･分散システムのモデル

2.2 並列プログラミングの形態

図1に示した並列･分散環境は既に身近なものとなりつつあるが、オペレーティングシステムとしてUNIXを採用し、逐次プログラミング言語であるCを用いる限り、信頼性の高い並列プログラミングを行うことは望めない。何故ならば、そのプログラミング環境では、図2(a)に示すように、依然個々のプロセッサ用に個別にプログラムを記述しなければならないからである。



(a) 各計算機毎に個別にプログラムを作成する



(b) 全体を一つのプログラムとして作成する

図2　並列プログラミングのモデル

これでは仮にAdaのような並列処理機能を持ったプログラミング言語を用いたとしても、その機能を活かすことができない。例えば、プログラム内ではランデブを用いることができても、プログラム間ではオペレーティングシステムのシステムコールを呼びださなければならない。従って、コンパイラは各

プログラム間のデータ授受の一貫性を検査できない。また、大域的な最適化を行うこともできないので、効率向上の機会も減る。

これらの問題点を解決し、参照の透明性を確保するためには、図2(b)に示すように、並列プログラミング言語を用いて、一つの並列プログラムとして記述することが必要となる。即ち、多重プロセッサ上の並列プログラムとワークステーション上の並列プログラムを同一の言語で一様に記述できなければならない。

2.3 プログラムの分割と割当て

全体を一つのプログラムとして作成することが可能になると、プログラムをある単位で分割し、各プロセッサへ割当てることを考慮しなければならなくなる。単一プロセッサ上で動作する並列プログラムが変更なく多重プロセッサ上でも動作することが望ましい。しかし、変数の共有やポインタによる参照を許すプログラミング言語では、言語仕様に制限を加えることが一般に行われている[2]。例えば、対象とする計算機システムが共有メモリを持たない場合、言語仕様では許されているポインタに使用を、プロセッサ間に跨る場合だけ禁止してしまうのである。

それでは並列プログラムが対象とするシステムのみならず、そのプログラム分割と割当てにまでに依存したものとなってしまう。プログラミング言語における参照の一様性は、そのプロセッサ間結合によらず保持されなければならない。また、共有メモリを持たないワークステーションを結合した粗結合分散システムは並列プログラミングを行う実験環境としても有望である[9,10]ので、これらの上でも密結合多重プロセッサと同様なプログラミング環境が望まれる。

3. 並列プログラミング言語ParaDisE

ParaDisE(Parallel/Distributed Environment) [3, 4]はメッセージ型通信機能とメモリ共有型通信機能を併せ持つAda言語[1]サブセット処理系である。単一アドレス空間の下でタスク、ランデブおよび共有変数を実現する。数値計算に代表される定型的な並列計算を負荷分散で高速化すると共に、粗結合･密結合入り交じる環境上で一様な並列プログラミング環境を提供する。

現在、ParaDisEの実行環境は粗結合の分散環境であるが、我々の最終目標は密結合と粗結合の混在した並列･分散環境上での並列アプリケーションの記述である。従って、粗結合の分散環境上では効率よりも記述の容易さに主眼を置き、密結合の並列環境上で効率の良い実現法を採る。

3.1 言語仕様

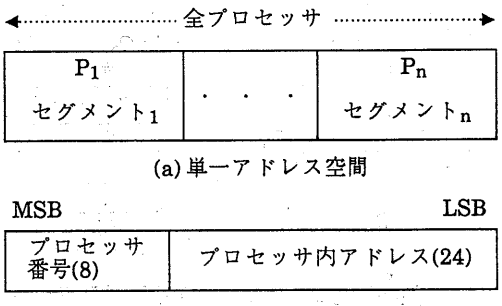
ParaDisEの言語仕様は並列プログラミング言語Adaを基にしている。並列処理に関わる機能の多くは取り込んだが、実時間処理に関する機能は削除した。サブセットであるため、既存の単一プロセッサ用処理系をプログラムの開発･テスト･デバッグ環境として利用できる。

Adaから採用した基本的な言語仕様は、パッケージ、配列、多重定義、アクセス型等である。削除した基本的な言語仕様は、汎用体、例外処理、密封型、レコード型、分割コンパイル等である。

Adaから採用した並列処理に関する言語仕様は、タスク(型)宣言、タスク配列、動的タスク生成、エントリ呼びだし、(選択的)アクセプト文等である。削除した並列処理に関する言語仕様は、エントリ族や条件付･時限付のエントリ呼び出し、ディレイ文、アボート文等、主に実時間処理に関するものである。また、タスクの従属関係および終了･消滅に関するセマンティクスの簡略化を行った。削除･簡略化した機能の大部分は、基となったAdaの基準文法書[1]に明確な定義がないか、分散環境上での実現が非現実的なものである[2]。

3.2 単一アドレス空間モデル

プログラムの任意の分割を許し、その分割間での共有変数およびポインタ(Adaではアクセス型)の使用を可能とするため、単一アドレス空間モデルを採用する。図3はそのイメージである。



(a) 単一アドレス空間

(b) 分散アドレス(32bit)

図3 単一アドレス空間モデル

単一アドレス空間モデルは、分散環境のアドレス空間を共有メモリのようなどこからでも見えるアドレス空間と考えるものである。この参照の一様性を保証するための単一のアドレスを分散アドレスと呼ぶ。密結合の並列システムでは分散アドレスとして

共有メモリのアドレスをそのまま用い、粗結合の分散システムでは分散アドレスへの参照をオペレーティングシステムでシミュレートする。論理的なアドレス空間のイメージと実際の並列･分散環境の物理的なアドレス空間のイメージとを分離しているので、任意のプログラム分割が可能となり、コンパイラは密結合･粗結合の差異に依存しないコードが生成できる。

3.3 局所参照と遠隔参照

対象とする並列･分散システムのメモリ参照には局所参照と遠隔参照の区別がある。従って、コンパイラは局所参照と遠隔参照の区別を付けることで、無用なアドレス変換のオーバヘッドを避ける。即ち、コンパイラはプログラムの静的解析によって、必ず局所参照になるか、あるいは遠隔参照になる可能性があるかの区別を付け、必ず局所参照になる場合は、分散アドレスの代わりに局所アドレス(プロセッサ番号0の分散アドレスを特別に局所アドレスと呼ぶ)を用いる。

従って、局所参照は単一アドレス空間モデルのオーバヘッドを受けない。ただし、局所アドレスを引数として渡す場合は、分散アドレスへの変換が必要となる。一般に、コンパイラが参照の局所･遠隔の区別を付けることで、アドレス変換を省略できる利点を生むと共に、分散キャシングの為のヒントをコンパイラが出力できる。

3.4 ランデブと共有変数

図4はAdaの基本的なランデブの制御の流れである。ランデブの引数はメッセージ通信の方法でも渡

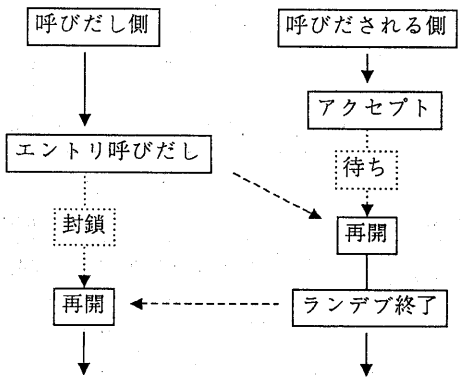


図4 ランデブ

すこともできるし、アドレス渡しでも構わない[1]。しかし、ポインタ型も引数として有り得る。従って、スカラ型等を除いて、分散アドレスで渡し、実際にその引数の内容に参照が起こる場合にはキャッシングを行って、それ以後の参照のコストを低減する。

Adaでは静的な変数の共有も可能だが、手続きやタスク内でスタック上に確保された変数も共有変数となる。この場合、動的な呼びだし関係のリンクを辿る必要がある。一般に入れ子構造を持ち、再帰呼びだしを許すプログラミング言語では、スタック上に活動記録を残し、そのリンクを辿ることで、スタック上に確保された変数への参照を実現している[11]。しかし、分散環境上では、参照の度にプロセッサに跨ってリンクを辿ることは著しい速度の低下を来す。ParaDisEでは、ディスプレイ法[12]を拡張した分散ディスプレイ法[3]を用いて、共有された動的変数への参照を高速化する。図5に分散ディスプレイ法の例を示す。このディスプレイにも分散アド

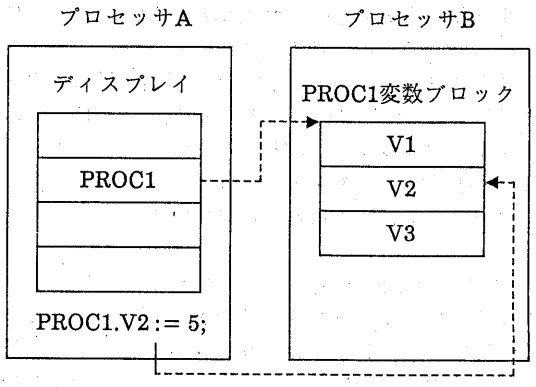


図5 分散ディスプレイ法

レスを用い、コンパイラは遠隔参照と局所参照の区別を付ける。

3.5 分割と割当て

単一アドレス空間モデルの採用によって、任意の分割が可能となったけれども、あまり細かな分割は割当ての際に煩わしさを生む。ある程度論理的にまとまった単位で分割を行うことが望ましい。ParaDisEでは、Adaにおけるタスク･パッケージ･手続き･関数等のコンパイル単位を分割の単位とした。

表1 属性

属性名	割当て
all	すべてのプロセッサ
any	何れか一つのプロセッサ
floating	必要(参照)の応じて
specified	プロセッサ番号指定

分割の際には、それぞれの単位に、表1に掲げる

属性のいずれかを付与する。タスクの大部分はどのプロセッサ上にも動的に生成され得るので、すべてのプロセッサ上にその分割を割当てるallの属性を持つ。他の単位は原則として、何れか一つのプロセッサ上にだけ割当てられるanyの属性を取る。ただし、以下に述べる例外がある。

複数のプロセッサから呼びだされる手続き·関数を、呼びだし側プロセッサ上で重複して、局所的に実行すれば、並列性を向上できることが多い。この様な処理形態を支援するために、ParaDisEにはfloating属性がある。コンパイラはそれ自身が遠隔参照を行わないようなパッケージ·手続き·関数を見つけだすと、それにfloating属性を付与する。floating属性を持つ分割単位への参照はすべて局所参照となる。即ち、参照する各々のプロセッサ上にfloating属性を持つ分割単位のコピーが配置されることになる。図6に割当ての例を示す。プログラムライブラリとして用意するパッケージ·手続き·関数の大部分はfloating属性を持つ。

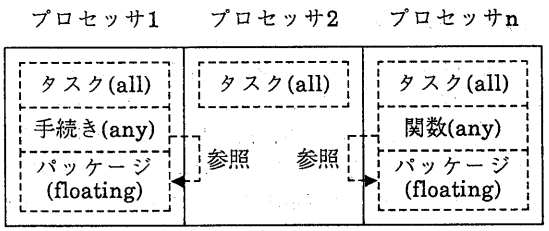


図6　割当ての例

従って、ParaDisEにおける手続き·関数呼びだしは、それら呼びだされる手続き·関数がany属性を持つならば遠隔呼びだしとなり、floating属性を持つならば局所呼びだしとなる。即ち、通常の手続き·関数呼びだしが、割当てに依存して、遠隔呼出しとなる。属性の付与はコンパイラが行い、その属性に依存して、遠隔参照·局所参照の区別および参照の最適化を行う。なお、specifiedはプラグマ等でプロセッサ番号を指定する場合に使用する。

3.6. 動的タスク生成

Adaのタスクは動的に生成することができる。all属性を持つタスクのコードは全プロセッサへロードされるので、動的生成の際にコードを転送する必要はない。どのプロセッサが選ばれるかは、オペレーティングシステムの責任である。一方、コンパイラは遠隔参照を減らすために、タスク生成の際に幾つかの情報を一緒に送る。定数はコードと一緒に置かれるが、それ以外にAdaには読み出し専用の変数や配列の上限·下限等の属性がある。これらは転送しなくても分散アドレスにより必要に応じて参照可能であるけれども、タスク生成時に送っておけば、プロセッサ間通信の低減につながる。

3.7. モニタリング機能

プログラム分割と割当てを、コンパイラやオペレーティングシステムが自動的にやってくれるのが理想であるが、実際には難しい。コンパイラの静的解析では不十分な部分を実行時の情報収集によって補う。モニタリングの目的はプログラム分割と割当てであるから、それに必要な情報だけを収集する。

ParaDisEでは、ランデブおよび遠隔変数参照の通信頻度·通信量をモニタリングする。モニタリングの計測値は各プロセッサ上で蓄積し、プログラム終了後に変数·エントリ名毎に回収·集計する。最終的に得られる計測値は、(1)各参照地点毎、(2)各変数·エントリ名毎、(3)各プロセッサ間毎、それぞれの通信頻度と通信量である。

3.8. 処理の流れ

図7にParaDisE処理系の処理の流れを示す。ParaDisE/Compilerが分割生成したオブジェクトをその属性に従ってParaDisE/Linkerでリンクする。それをParaDisE/Loder(現在これはDaOS内にある)が各プロセッサ上へロードし、DaOSがそれを実行する。

4. 分散オペレーティングシステム核DaOS

DaOS(Distributed Ada Operating System)[3,5]は並列プログラミング言語ParaDisEで記述された並列プログラムの実行を支援する分散型オペレーティングシステム核である。並列プログラムとのインタフェースとなるランデブ等の並列プログラミング言語特有の機能を提供すること、およびParaDisEとのインタフェースにおいて前提となっている単一アドレス空間モデルを実現することによって、並列プログラムを分散環境上で実行することを支援している。

4.1 基本並列処理機能

図8はUNIX上に構築したDaOSの構成である。メモリ管理·スレッド管理·入出力管理は、XINU[6]を基にしており、単一プロセッサ内の仕事を行う。ネットワーク管理は、UNIXのTCP/IPプロトコルを利用して、各プロセッサ間の通信を受け持つ。

分散アドレス管理は単一アドレス空間を上位層に提供しする。この層が提供する基本的な機能は、遠隔読み出し·書き込み、封鎖·非鎖型遠隔手続呼びだしである。タスク管理は、他のプロセッサ上でのタス

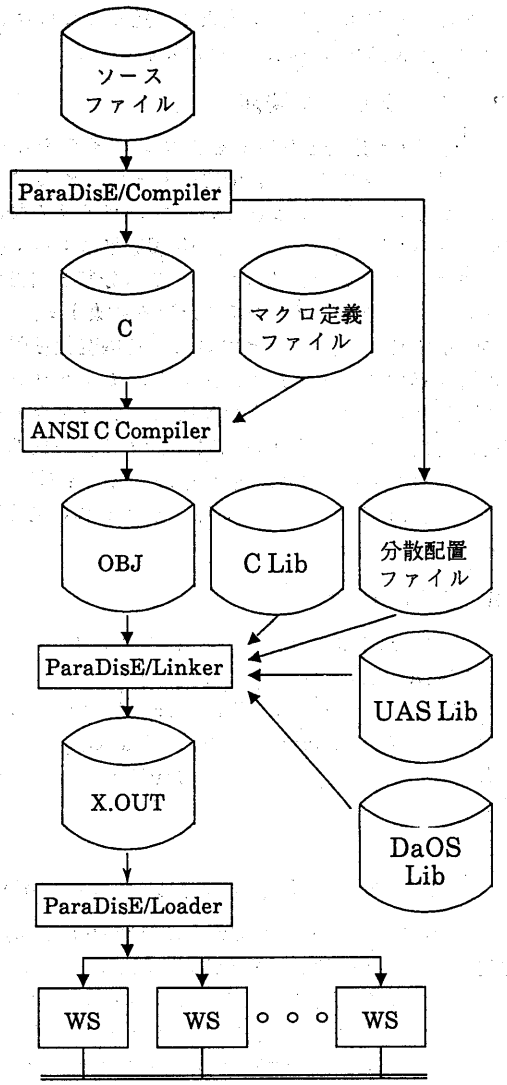


図7　ParaDisE処理系の処理の流れ

ローダ	並列アプリケーション	
	インタフェースライブラリ	
分散アドレス管理	ランデブ	タスク管理
ネットワーク管理		
メモリ管理	スレッド管理	入出力管理
UNIXカーネル		

図8　DaOSの構成

クの生成･消滅およびAda言語特有のタスクの親子関係を管理する。

ランデブは、分散アドレス管理とタスク管理の境界に位置する。ランデブ層が提供する機能は、単純なエントリ呼びだしと単純なアクセプトおよび選択的なアクセプトである。この層が提供するランデブの引数は一つでアドレス型である。そのアドレス型引数を用いて言語とのインタフェースライブラリが間接指定することによってAdaにおける3種類のモードでの引数渡しを実現している。

タスク管理はタスクの生成･消滅と共に、親子関係を管理し、Ada独特のタスクの従属関係および生成･消滅に関するセマンチックスを実現する。

ローダはDaOS内の一つのカーネルスレッドとして動作する。ローディング時にプロセッサ番号を決定し、各プロセッサに跨る外部参照を解決する。その際、図6で示したように、all属性を持つものからロードすることで、タスクのプロセッサ内アドレスがすべてのプロセッサ上で同一になることを保証する。なお、動的なタスクの移動は行わないので、ローディング以後、その分散アドレスは変わらない。

4.2　スレッド

DaOSのタスクはアドレス空間を共有するスレッド(軽いプロセス)である。DaOS内の通信にもこのスレッド機能を用いる。遠隔読み出し･書き込みは相手プロセッサの通信サーバ(カーネルスレッド)と協同して行う。一方、遠隔手続き呼びだしは封鎖されるので、呼びだし側でエージェント(カーネルスレッド)を作成し、これに代行させる。図9にスレッドを使用したランデブの実現を示す。同一プロセッサ上のカーネルスレッド同士ではセマフォ同期を用いる。

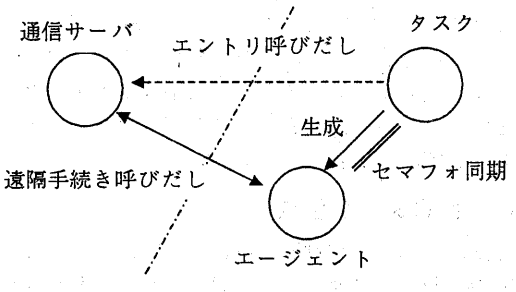


図9　ランデブの実現とスレッド

UNIX上では、これらスレッドの集合から成るDaOSプログラムが一つのUNIXプロセスであり、仮想的なプロセッサに相当する。図10に示すように、一台の物理プロセッサ上に複数の仮想プロセッサを割当てることができるので、柔軟なプログラム構成を可能にしている。

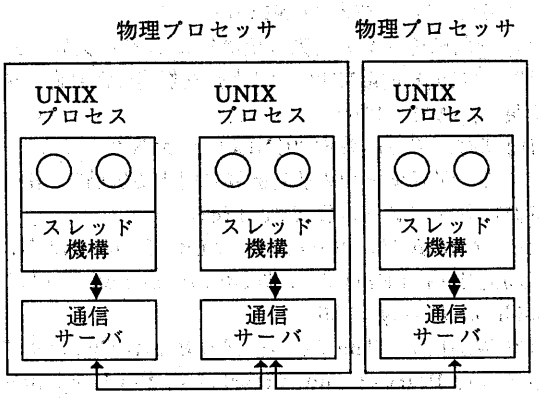


図10　UNIX上での実行

5. 評価と考察

ParaDisEの処理系はDEC社のVAXStation上で稼働している。また、DaOSは30台のVAXStationがLANで結合された分散環境上で動作する。開発した処理系上でテストプログラム作成を行い、実際の分散環境上で動作させた結果、記述上のおよび実現上の問題が見つかった。

5.1　タスク配列と放送型ランデブ

プログラム中で、利用できるプロセッサの数を使いたい。Adaでは提供されていないが、読み出し専用の変数として参照できることが望ましい。これを使えば、実行時に適当な大きさのタスク配列を動的に生成することが容易になる。

また、タスク配列のインデックスを得るための機能も欲しい。Adaでは配列となっているタスクが自分のインデックスを知ることができず、親タスクとのランデブを行わなければならなかった[2]。

繰り返してタスク配列全体に同じランデブを行う場合、放送型のランデブを行いたい。タスク配列に対するランデブは良く使われるのだけれども、ランデブは待機するので、多くのタスク配列に対して対してランデブをすると時間を消費してしまう。

```
for I in TASKS'RANGE loop
    TASKS(I).E(DATA);
end loop;
```

上の例では、同じランデブを各タスクに行うので放送型ランデブを使いたい。

5.2　基本操作の速度

10Mbpsのイーサネットで接続された2台のVAXStation3500(約3MIPS)上で、ParaDisE/DaOSの基本操作の速度を求める測定を行った。プログラムはParaDisEを用いて記述した。表2はその結果である。それぞれのデータサイズに対する基本操作1回に要した経過時間である。非同期入出力の採用、TCPの詰め込み禁止などで、以前[14]に比べると数倍の速度向上が実現できた。

表2　基本操作の速度

操作	データサイズ (バイト)	経過時間 (ミリ秒)
遠隔読み出し	4	13
〃	1024	18
遠隔書き込み	4	7
〃	1024	10
ランデブ	0	23
〃	1024	37

読み出しが書き込みに比べると経過時間が長いのは、書き込みがその要求とデータとを同時に送れるのに対し、読み出しはその要求とデータとを2回に分けて送らなければならないからである。ランデブには遠隔読み出し1回に遠隔書き込み1回を加えた程度の通信が必要で、その上にタスクの起動によるオーバヘッドが加わっている。

表3　ランデブ(付録A)の速度比較

システム名	データサイズ (バイト)	経過時間 (ミリ秒)
ParaDisE/DaOS	4	44
SUN RPC	4	10

次に、整数を一つ受取り、それに1を加えて返すランデブ(付録参照)を実行した。また、付録と同様の処理をSUN Microsystems社のRPC[7]を用いてC言語で記述して実行した。表3はその結果である。ParaDisE/DaOSが1回の呼びだしに44ミリ秒必要としたのに対し、SUN RPCは10ミリ秒しか必要としなかった。この比較から、DaOSにはParaDisE支援のためのオーバヘッドがあるので、ほぼ同程度の所要時間とみなせるものの、DaOSの構成、特に階層構造にまだ改良の余地があることが分かった。

6. むすび

本稿では、メッセージ型通信機能とメモリ共有型通信機能を併せ持つAda言語サブセット処理系ParaDisE、およびParaDisEで記述された並列プログラムの実行を支援する分散オペレーティングシステム核DaOSの実現について述べた。今後は、分散環境上で通信プロトコルの最適化、キャッシング、自動プロセッサ割当て等について検討を続けると共に、現在稼働している両システム上での並列アプシケーションの記述を通して、新しい並列プログラミング言語の設計へと進んで行きたい。既に稼働中の処理系を使って並列アプリケーションを幾つか記述し、分散環境上で動作させたが、その結果については別稿で報告したい。なお、両システムは、最終的な対象を九州大学総合理工学研究科富田研究室の可変構造型並列計算機[8]としている。

謝辞

九州大学総合理工学研究科の福田晃助教授、九州大学情報工学科の吉田紀彦助手には並列処理に関して幾度も議論に加わって頂いた。日頃から厚く御教示頂く九州大学情報工学科牛島和夫教授に感謝致します。

参考文献

[1] Ada Programming Language (ANSI/MIL-STD-1815A), U.S. Government, Department of Defense, Ada Joint Program Office, 1983.

[2] Burns,A., Lister,A.M., and Wellings,A.J.: "A Review of Ada Tasking," Lecture Notes in Computer Science, Vol.262, 1987.

[3] 岡村、縄田、平原、荒木: "単一アドレス空間モデルに基づいた分散環境上での並列プログラミング言語処理系の実現"、信学技報、CPSY89-20、pp.33-38、1989.

[4] 縄田、岡村、平原、荒木; "並列/分散環境上でのプログラミグ言語処理系ParaDisE"、情報処理学会第40回全国大会講演論文集、pp.754-755、1990.

[5] 岡村、縄田、平原、荒木: "分散環境上での並列プログラミング言語処理系用のOS DoOSの概要"、情報処理学会第40回全国大会講演論文集、pp.756-757、1990.

[6] Comer,D.: "Operating System Design, the XINU Approach," Prentice-Hall, 1984.

[7] SUN Microsystems.: "Remote Procedure Call Programming Guide," SUN OS Manual, 1987.

[8] 村上他: "可変構造型並列計算機のシステム・アーキテクチャ"、情報処理学会「コンピュータ・アーキテクチャ」シンポジウム論文集、pp.165-174、1988.

[9] 岡村、平原、荒木: "UNIXワークステーションによる分散環境上での並列数値計算に関する実験と評価"、平成元年度電気関係学会九州支部連合大会講演論文集、pp.507、1989.

[10] 岡村、平原、荒木: "UNIXワークステーションによる分散環境上での並列計算に関する実験"、第14回JUSシンポジウム論文集、1989.

[11] Aho,A.V., Sethi,R., and Ullman,J.D.: "Compilers, Principles, Techniques, and Tools," Addison-Wesley, 1986.

[12] Wirth,N.: "The Design of a Pascal Compiler," Software-Practice and Experience, Vol.1, pp.105-133, 1971.

[13] 縄田: "並列/分散環境上でのプログラミグ言語処理系ParaDisEの実現"、九州大学大学院工学研究科修士論文、1990.

[14] 岡村: "粗結合分散環境上での並列計算に関する研究"、九州大学大学院工学研究科修士論文、1990.

付録 - テストプログラム

```
function CLOCKM return INTEGER;

with TEXT_IO;
use TEXT_IO;
procedure MAIN is
  package INT_IO is
    new INTEGER_IO(INTEGER);
  use INT_IO;
  ITERATION: constant := 1000;
  P: INTEGER := 1;
  T1, T2: INTEGER;

  task SIMPLE is
    entry INC(P: INTEGER; Q: out INTEGER);
  end INC;

  task body SIMPLE is
  begin
    while TRUE loop
      accept INC(P: INTEGER; Q: out INTEGER);
        Q := P+1;
      end INC;
    end loop;
  end INC;

begin
  T1 := CLOCKM;
  for I in 1..ITERATION loop
    SIMPLE.INC(P, P);
  end loop;
  T2 := CLOCKM;
  PUT("TIME=");
  PUT((T2-T1)/ITERATION);
  NEW_LINE;
end MAIN;
```