解　説

並列プログラム処理実験装置†

高橋義造†† 吉村　晋†††

1. はじめに

並列処理の一つの型である MIMD (multiple instruction stream-multiple data stream) はマルチプロセシングとも呼ばれ，2～4 の並列度のものは大型計算機においてすべて実用されている．大型計算機におけるマルチプロセシングは，マルチプログラム処理を目的としている．すなわち，それぞれのプロセサが何個づつかのプログラムを分担して同時処理するようになっている．一方最近ではマイクロプロセサのような小型の cpu を多数使用すれば数十乃至数百の並列度を得ることが難しくなくなってきた．そこで多数のプロセサを使用して単一のジョブ又はプログラムを処理することにより処理速度の向上をはかろうという試みが現われた[1),2)]．この方式を並列プログラム処理と呼ぶことにする．

並列プログラム処理によって計算速度の向上をはかるため取組まなくてはならない課題は次の二つである．第一は，一つのプログラムをどのように分割して各プロセサに分担させ，処理時間を短縮するかという課題であり，このためには効率のよい並列処理アルゴリズムを開発しなくてはならない．プログラムの一部を分担しているプロセサが，他のプロセサとできるだけ干渉しないようなアルゴリズムでないと，折角多数のプロセサを使いながらプロセサ間の干渉に時間を空費して処理時間が短縮できないばかりか，プロセサ数を増すに従って処理時間が増加するようなことになりかねない．

第二の課題は，プロセサ間の交信と共有データへのアクセス競合によって生ずるオーバヘッドを極力減少するようなハードウェアと OS の方式を開発すること

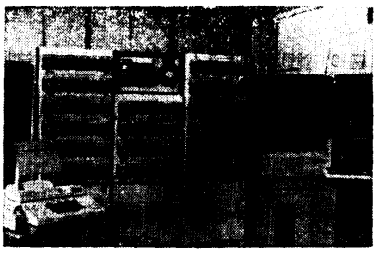

図-1　並列プログラム処理実験装置の外観

である．

これらの課題に取組むためには，各種のアルゴリズムやハードウェアの方式について処理速度を測定する必要がある．このため我々は先づシミュレーションによる研究を行った．当初は電子技術総合研究所で開発された PPSS[3)] を使用させて頂き，ついでこれをベースにして開発した IPPSS[4)] を使用した．この結果ある程度の見通しをえたので，OS を含めた実際の環境の下で，シミュレーションでは扱えなかったような並列プログラム処理の実験を行うために「並列プログラム処理実験装置」と呼ばれるシステムを開発し，OS を試作した．この装置は昭和 52 年 10 月に完成し，その後性能向上のための改造，周辺機器の追加を行い，現在は図-1 の外観のようなものになっている．

2. システム構成

並列プログラム処理実験装置のシステム構成を図-2 に示す．cpu として 16 ビットマイクロコンピュータ TOSBAC-40 L (16 ビット加減算 2.7 μsec) のワンボード cpu を使用した．cpu は 16 台あり，各々に 0～15 の cpu 番号をつける．この cpu 番号は cpu ごとにあるイニシャルプログラムローダの ROM に書かれており，cpu を起動するとローカルメモリの特定番地にこの番号が書き込まれるようになっている．第 0 番目の cpu をコントロールプロセサと呼び，他の 15 台の cpu をワーキングプロセサと呼ぶ．16 台の cpu にはそれぞれ 32 KB のローカルメモリ（サイクルタ

† A Test Equipment for Parallel Program Processing by Yoshizo TAKAHASHI (Tokushima University) and Susmu YOSHIMURA (Toshiba Research and Development Center).

†† 徳島大学工学部
††† 東芝総合研究所

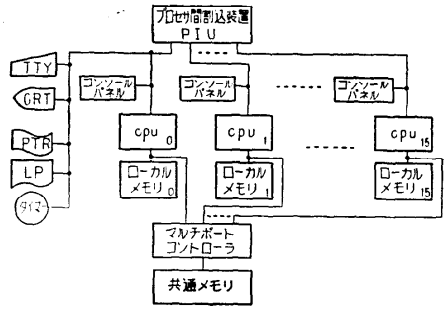

図-2 並列プログラム処理実験装置の構成図

イム 0.8 μsec) が接続されている．各 cpu のアドレス空間の最初の 32 KB がローカルメモリに割当てられる．

本来ならばシステム全体の制御と監視のためにシステムコンソールを設けるところであるが，この装置では便宜上 TOSBAC-40 L の標準のコンソールパネルをそのまま使用し，筐体の前面にこれを 16 台並べて取付けることとした．

この実験装置では cpu 間の通信のためにプロセサ間割込装置 (PIU) と共通メモリの二つの装置が用意されている．PIU は全 cpu の入出力バスに接続され，すべての cpu は自分自身を含めて任意の cpu に割込み信号と，mode とよぶ3ビットの情報を送ることができる．割込まれた cpu は PIU のステータスを調べることにより，割込んで来た cpu の番号と mode を知ることができる．

共通メモリは 32 KB の記憶容量をもつ 16 ポートのメモリ (サイクルタイム 0.4 μsec) で，16 台の cpu のメモリバスにはローカルメモリと，共通メモリの一つのポートが並列に接続されている．共通メモリには各 cpu の 32 KB～64 KB のアドレス空間が割当てられている．

この装置の入出力装置としては，入出力タイプライタ，ディスプレイ，紙テープリーダ，ラインプリンタ，タイマーが各1台づつあり，これらは一括してコントロールプロセサの入出力バスに接続されている．従ってワーキングプロセサがこれらの入出力装置を使用する場合には，コントロールプロセサにメッセージを送って入出力を依頼し，共通メモリを用いてデータを転送する必要がある．これは OS によって自動的に行われる．

3. 動作の概要と処理方式

この実験装置で実行される並列処理のためのプログラムは，一つの **program** と，いくつかの **process** より構成される．**program** は主としてプログラム全体の制御を行う部分で，これは必ずコントロールプロセサによって実行される．**process** は **program** がシステムマクロ activate をコールしたときに起動されるタスクで，ワーキングプロセサで実行される．通常は並列処理のために同じ **process** が別々のローカルメモリにローデイングされ，同時に実行されるが，**process** は一種類とは限らないので，異なる **process** が，異なる cpu で同時に実行されることもある．

program や **process** を管理し，プロセサ間通信や入出力機器の制御を行うための各種のシステムマクロを解釈，実行するために PARALLEL と呼ぶ OS を開発した[5]．PARALLEL の下では cpu 間の相互通信はシステムマクロ pcom を用いて行う．pcom には 0, 1, 2, 3 の4つのモードがあり，モードが0と1のときはメッセージの転送をともない，2と3のときはともなわない．モードが0のときは共通メモリにおかれたメイルボックスの内容そのものがメッセージであり，モードが1のときはメイルボックスにメッセージのサイズと格納場所が入っていることを示す．モード2は送られてきたメッセージに対する ack の応答，モード3は nack の応答になる．

4. 並列処理プログラム

我々は並列処理プログラムを記述するために PPL (parallel processing language) なる言語を考え，使用している．PPL の言語仕様の原形は文献[6]にあるが，現在はこれを拡張したものについてコンパイラを開発中である．PPL では並列処理プログラムは，データ宣言部，process 部および program 部よりなる．データ宣言部は program 部や process 部の中で使用されるデータのうち共通メモリにおくものを定義する部分である．

process 部ではワーキングプロセサで実行されるプログラムの手続きと，ローカルメモリにおかれるデータの記述を行う．program 部ではコントロールプロセサで実行されるプログラムの手続きと，そのローカルメモリにおかれるデータの記述を行う．

プログラムは先づコントロールプロセサによる

program の実行より始まり，これが activate 文を実行するとワーキングプロセサが起動され，**process** が実行される．activate 文は次のようなものである．

例. activate s, input(1), compute(3, 4, 5);
　　output(8);

ここに s はカウンティングセマフォアであり，この文により起動された cpu の台数に等しい数（この場合は5）が s に加算される．input, compute, output などは **process** の名前であり，カッコ内の数字はその **process** を実行する cpu の番号である．activate 文によって起動された **process** は処理を終ると cause 文を実行する．

例. cause　s, 1;

この例ではカウンティングセマフォアが1減じられる．

一方 activate 文を実行したコントロールプロセサは次に await 文を実行することにより，セマフォアが一定数に減少するまで待機する．

例. await s, 0

この例では s が0になるとはじめて次の文の実行に移ることが可能になる．この外 cpu 間の相互排除の制御のために lock 文，unlock 文が使用される．PPL の使用例は文献[6]に譲る．

5. 使用例と考察

本実験装置を用いて，ベクトルの内積 $\sum_{i=1}^{1024} a_i b_i$，テーブルサーチ，最大値の探索，$\sum_{i=1}^{5120} a^n{}_i$，Gauss-Jordan 法による逆行列の計算等の計算の並列処理を行った．ワーキングプロセサの台数を1台から 15 台までかえて処理速度の向上率を測定した結果を**図-3** に示す．

つぎに内積計算のプログラムを用いて共通メモリに対するアクセス競合[7]，相互排除機能による影響，およびオペレイティングシステムのオーバヘッド等が並列処理の効率に及ぼす影響を考察する．

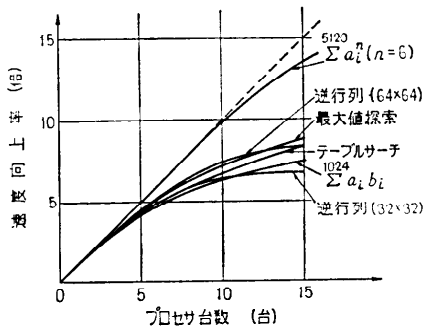

図-3　並列処理プログラムの速度向上率

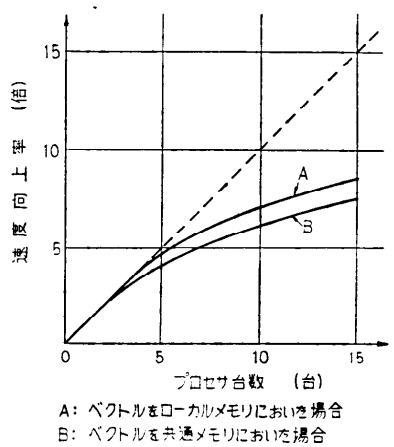

図-4　ベクトルの内積計算における共通メモリへのアクセス競合の影響

マルチプロセサにおける共通メモリへのアクセス競合による処理速度の低下の問題は TSS におけるユーザの think time と計算機の service time に基づく response time の関係と等価である．並列処理速度を向上させるためには

（1）共通メモリに高速メモリを用いると同時にポートの切りかえを高速化して service time を減少させる．

（2）think time を増すようなアルゴリズム，すなわち極力共通メモリにアクセスしないようなアルゴリズムを開発すること，

図-4 は，ベクトルの内積計算の場合にベクトルをローカルメモリに重複しておいた時と，共通メモリにおいたときの処理速度の比較を行ったものであるが，大きな差はない．従って共通メモリへのアクセス競合の影響は 10% 程度であるといえる．

相互排除機能による影響は並列度が大きくなると急激に増加する．**図-5** はベクトルの内積計算に当ってベクトルの一要素をよむ度に test and set を行った場合と，test and set の使用を最小限に止めた場合の速度向上率を比較したものである．

我々の試作した OS はできる限り cpu 間の干渉を少なくするように構成されているが，やはりいくばくかの干渉はさけられず，これが処理速度の向上を妨げているものと思われる．そこでベクトルの内積計算を OS なしで実行したものと，OS の下で実行したものを比較した．これを **図-6** に示す．OS により cpu が

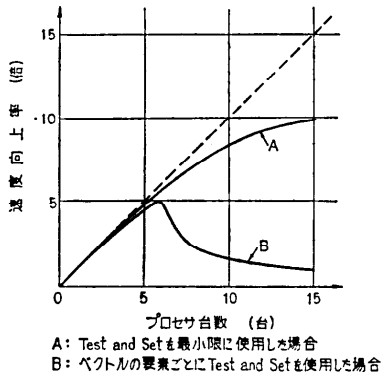

図-5 ベクトルの内積計算における相互排除機能の影響

15台のとき約30%向上率が減少するが，OSのある場合はcpu台数を示したときの向上率の飽和がおそいことがわかる．

6. おわりに

数十台～数百台のプロセサを接続して高速演算を行うような並列処理システムを目標に本実験装置を開発し，いくつかの基本的な実験を行った．その結果，これまでに得られた感想は，この実験装置のような一元的なアーキテクチャではプロセサ台数がせいぜい16～32台程度が限界であり，それ以上プロセサを増しても処理速度の向上は困難であるということである．MIMDだけではなく，これにMISD等異種のアーキテクチャを組合わせた新しい方式を導入する必要があろう．

本実験装置は，通産省工業技術院の大型プロジェクト「パターン情報処理システムの研究開発」の一環として開発されたものである．PPSSを使用させて頂いた電子技術総合研究所古谷立美氏をはじめ，本装置の製作に当り有益な助言を頂いた東芝総合研究所山崎勇氏，製作を担当された(株)日本ビジネスオートメーションの十市章弘氏，川野誠氏，ソフトウェアの作成に協力いただいた芝浦システム(株)岡林典明氏に感謝の意を表す次第である．

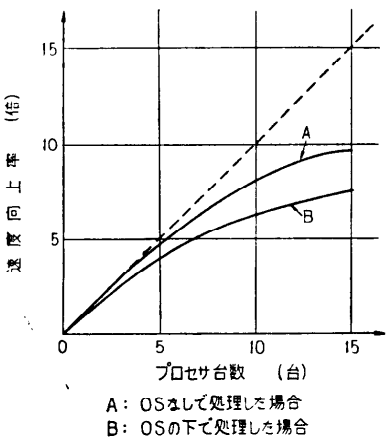

図-6 ベクトルの内積計算におけるOSのオーバヘッドの影響

参 考 文 献

1) Wulf, W., et al.: C. mmp—A Multi—Mini Processor, FJCC Vol. 41, pp. 765-777(1972).
2) Fuller, S. et al.: Multi Microprocessors: An Overview and Working Example, Proc. IEEE, Vol. 66, No. pp. 216—228(1978).
3) 古谷立美：ポリプロセサ・シミュレーションシステム—PPSS, 情報処理, Vol. 18, No. 6, pp. 534—541(1977).
4) 高橋義造，藤田純一：ポリプロセサのソフトウェアシミュレーションIPPSS, 第18回情報処理学会大会, 95(1976).
5) 高橋義造，吉村 晋：並列プログラム処理実験装置とそのOS, 情報処理学会アーキテクチャ研究会, 31-1(1978).
6) 高橋義造：並列処理プログラミング言語と並列処理アルゴリズム，第18回情報処理学会大会, 244(1977).
7) 坂東忠秋他：制御用マルチコンピュータシステムにおける共有メモリの設計と解析，情報処理, Vol. 19, No. 9, pp. 810—816(1978).

(昭和53年12月1日受付)