43

並列計算機の構築と数値流体解析

柳澤三憲[1)]、畠山伸[2)]

Construction of a Parallel Computer System and Its Applications for Numerical Simulation

By

Mitsunori Yanagizawa and Shin Hatakeyama

ABSTRACT

Construction of parallel computer systems for Intel and Alpha chips has been preformed in our laboratory. The former is make from all commercial parts i.e. Intel Celeron 300A CPUs, SD-RAM 128mb memory boards and a Linux OS. An Ethernet with a 100BASE-TX card that has a message rate of 100Mbit per second is used for communicating between each unit. The MPICH library uses a message-passing interface. We have already developed some kinds of parallel programs using the QR method, the LU decomposed method, and a finite volume method for use in a parallel computer. A part of this research was supported by Frontier Research Center for Computational Science in Science University of Tokyo.

---

1) 東京理科大学教授
2) 東京理科大学 M2

1. はじめに

フォン・ノイマンはコンピュータ時代の入り口を開いた。彼は、プログラム内蔵方式によって、現在の計算機の幕開けとなる一号機を製作した。その背景には核兵器の開発があった。最大の破壊力を生むには、どう核爆発を起こせばよいか。そのための膨大な計算が必要であった。ノイマンらが製作したコンピュータは高さ 1.8m、横 3m、奥行き 0.75m のものである。その後半世紀の間に個人で並列計算機が持てる時代になった。

従来の汎用スカラー計算機による数値計算では膨大な計算時間が通常費やされる。特に、三次元流体シミュレーションでは一昼夜ぐらい掛かる計算が非常に多いのであるが、この計算時間を四分の一ぐらいに短縮できれば計算プログラムの開発に有力な手段となる筈である。我々の研究室では並列計算プログラムを自宅で作成し、コンパイル出来れば、さらに開発速度が速くなる。そこで、Personal Parallel Computer(PPC)を目指して、低価格の PPC を構築し、そのマシンで開発した並列プログラムはそのまま本格的な並列計算機(IBM SP2)に掛かり、並列計算できることを目的としている。したがって、各ユニィットは汎用の AT 互換機を採用した。

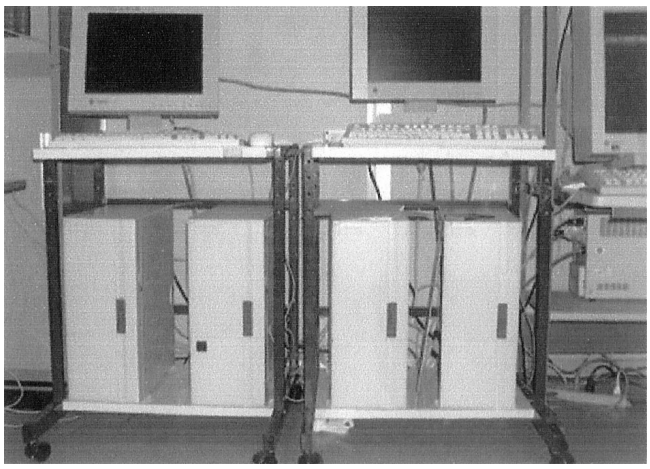

図 1　Intel 版並列計算機の概観写真

CPU としては Intel Celeron 300A, memory SD-RAM 128MB を用いて製作した。OS には Linux を用い、distribution として Plamo や RedHat を採用した。また、通信手段として Ethernet の 100BASE-TX を用いた。通信速度は

100Mbit/second である。また、ライブラリーとしてはMPICH(Message Passing Interface)をも用いて、相互通信を行っている。

並列計算プログラムはQR法、LU分解法、有限体積法による流体解析を開発した。また、新しいアルゴリズムで最初から並列化を考慮したZ-level Programming Language (ZPL)言語を使用して、これらのプログラムを完成させた。QR法ではFORTRAN 90,ZPL言語で作成した。有限体積法による流体解析ではC言語でMPIにより並列化を行った。

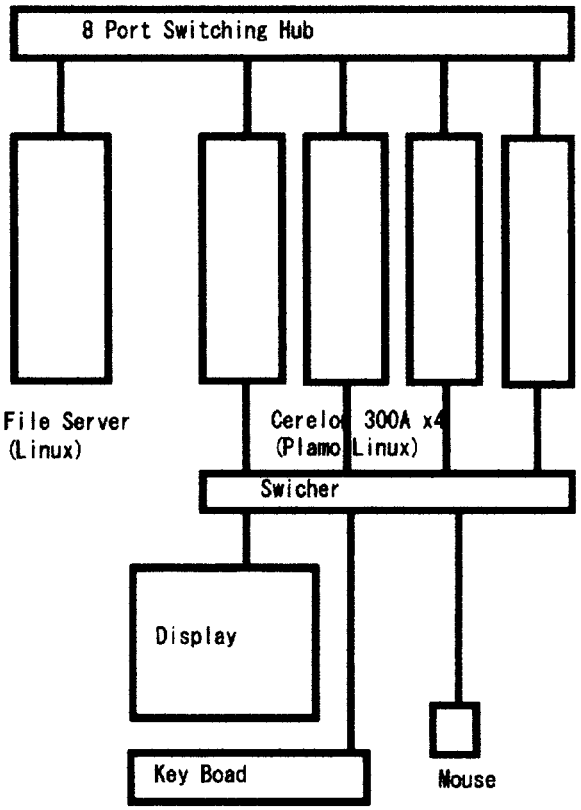

図2　並列計算機のブロック線図

2. 並列化の原理

並列処理が困難な処理対象において、プロセスの逐次処理がs (0<s<1)であるとすれば、n台のプロセッサを用いて得られる速度向上の割合は

$$1/(s+(1-s)/n)$$

で与えられる。これをアムダールの法則と言う。例えば、逐次処理の割合が5%のジョブがあったとして4台のプロセッサを使って3.47倍の処理速度が得られるが、1000台のプロセッサを使っても高々19.96倍程度の性能しか得られない。したがって、並列処理化する際には解くべき問題の性質をよく把握し、並列処理のノード数を検討する必要がある。画像処理のように時間的、空間的に独立性の高い処理対象のジョブではほぼ台数に比例した性能が得られる。

3. 並列計算機の構築

並列計算機の構築にはまず部品の選択が必要である。特にOSとしてLinuxを用いるため、マザーボードやCPU, Net card, hard disk, graphics boardなどがLinuxで認識できるパーツを選択しなければならない。

パーツ	AT互換機	Alpha CPU機
CPU	Intel Celeron300A	Alpha21164A
1次キャッシュ	32KB	8KB+8KB
2次キャッシュ	128KB	96KB
3次キャッシュ	non	4MB on board
CPU Clock	450MHz	600MHz
Mother board	ASUS P2B-D	AlphaPC 164LX
Memory	SD-RAM 512M	SD-RAM 512M
NIC	10/100BASE	10/100Mbps
HDD	E-IDE10.4GB	9GB Ultra Wide SCSI
Video Card	Trio64V2M	Milenium II
OS	Plamo Linux	RedHat Linux

表1　Intel版とAlpha版の部品の比較

4. 並列計算機の構成

その構成は汎用のAT互換機であり、一般に市販されているものである。当研究室で試作したAT互換機による並列計算機を図1に示す。すなわち、LinuxのOSで作動するファイル・サーバ、100Mbpsのスィッチングハブ、スィッチヤー、デスプレイ、キィーボード、マウスを配置した。このファイル・サーバはNIS(Network Information Service)が各ユニットに対応した環境変数がセットしてある。(図2。)　これによりMPI通信を適切に行う。また、本

格的に並列化した流体解析プログラムを走らせるため、Alpha Chips による並列計算機も製作した。これにより、三次元有限体積法によるキャビティー・フローの解析を行った。

5. NFS および NIS 設定

OS のインストールおよびセットアップについては、その情報が多岐にわたるため、基本システムはすべて Linux とする。セキュリティについては、パフォーマンスの低下を避け、管理の容易さを優先するために最低限のものにする。したがって、他のネットワークからは隔離する。不要なプロセスは極力排除しておく。

並列計算機による分散処理を支えているのは、主に rsh(リモートシェル)コマンドである。rsh コマンドを使うことで、現在ログインしているのとは違うホスト上でコマンドを実行することが可能となる。rsh コマンドを快適に実行するためには、OS の設定が必要である。

NFS は、複数のホスト上でディスク情報を共有するためのシステムである。並列計算ライブラリを利用したプログラムは、各ホスト上に同一のディレクトリ、ファイル構成を要求するので、NFS による各種ファイルの共有が推奨される。具体的に必要な設定は次の通りである。

① NFS サーバーはカーネルを NFS サービス対応に再構築する。
② デーモンは rpc.mountd, rpc.rfsd, inetd を起動する。
③ 設定ファイルは /etc/exports, /etc/hosts.equiv である。
④ NFS クライアント側では同様にカーネルを NFS サービス対応に再構築する。
⑤ 設定ファイルは/ect/fstab である。

並列処理におけるデータのどうきプロセスごとの情報交換に関して取り扱いやすくしたものが並列処理ライブラリである。現在の標準として最も活動が盛んな MPI ライブラリを用いた。フリーの配布が許されている MPICH を利用する。rsh の環境が整備され、/home ディレクトリの共有も済んでいるものとする。基本的には管理者のコンパイルした mpicc, mpif77, mpif90, mpirun などのコマンドを利用する。

5-1.簡単な導入方法

まず、MPICH のソースコードを入手する。

```
#tar zxvf mpich.tar.gz
```

で展開した後、mpich ディレクトリに移動し、

```
#./configure
#make
```

でコンパイルする。詳細はドキュメントファイルを参照のこと。必要な設定ファイルとして、/utils/machines の下になる

```
machines.LINUX
```

がある。これは、並列処理のプロセスを発注するホスト名を記載したもので、プロセスを発注する順にホスト名を列挙しておく。サンプルプログラムを動かすには、/examples/basic などへ移動して、

```
make cpi
```

とする。コンパイルした後、

```
mpirun –np 4 cpi
```

とする。但し、mpirun は、/bin 上にあるのでリンクを張っておく必要がある。ここで、-np 4 は 4 つのプロセスで動かすことを意味している。

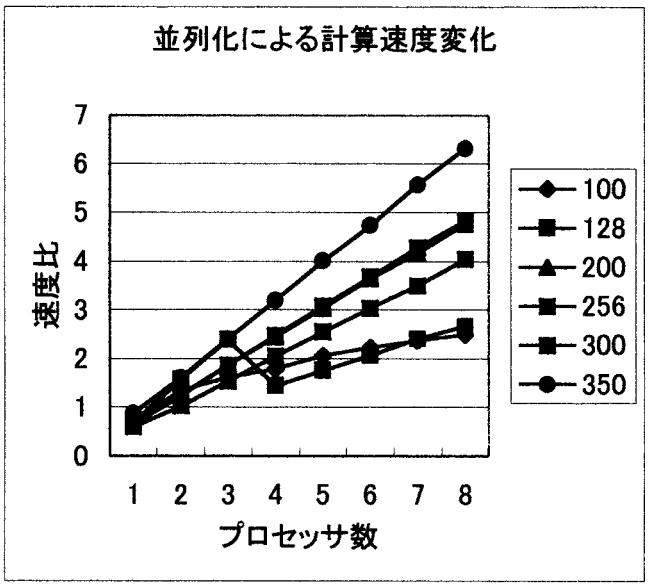

図 3 元数による計算速度

6. 計算結果

Intel 版と Alpha 版の並列計算機について、OS とし

て Linux を使用し、MPI を用いて FORTRAN 90 や C 言語で並列化プログラムを作成した。有限要素法による流体解析ソフト、QR 分解法、ピボット選択を行う LU 分解法などを作成した。特に、並列化した QR 法や LU 法について、ノード別の計算を行い並列化の効果を調べた。

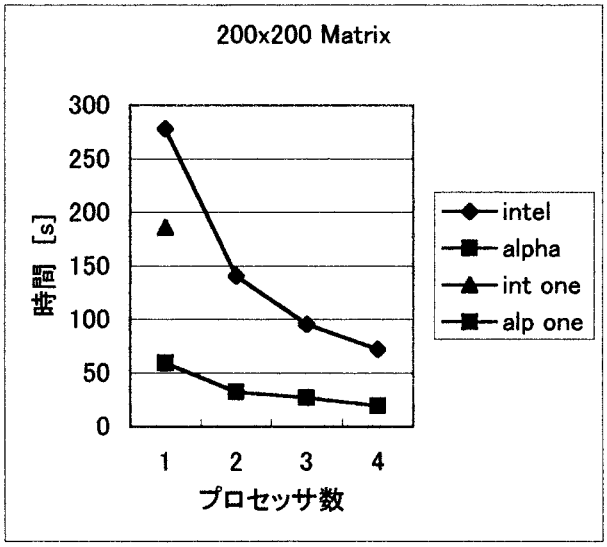

図 4　Intel 版と Alpha 版の計算時間

6-1. AT 互換機による計算結果

図 3 には並列化した QR 法について、ノード数に対する計算時間をプロットしたものである。マトリックスの一辺の元数を 100, 128, 200, 256 と変えて、並列化のプロセッサの数に対する計算速度を測定した。この縦軸の速度比の意味はスカラー計算プログラムによる計算時間対する並列化した計算プログラムによる計算時間を示したものである。したがって、プロセッサ数 1 の値は 1 以下になっている。これは MPI メッセージによる時間が加算されるためである。

元数が少ないと相互通信に時間を費われて計算速度は上がらない。元数が 200 以上ではプロセッサ数に比例して計算速度も増加していることが分かる。しかし、理想的な直線から見るとその勾配は小さい。

6-2. Alpha CPU 機を用いた計算結果

本格的に流体解析を行うにはやはり Alpha CPU を用いた並列計算機が最良である。(図 6。) AT 互換機での勾配の低さと比較して理想的な直線に近い結果が得られた。だだし、元数が 100 ではほとんどその効果が表れないことが分かる。

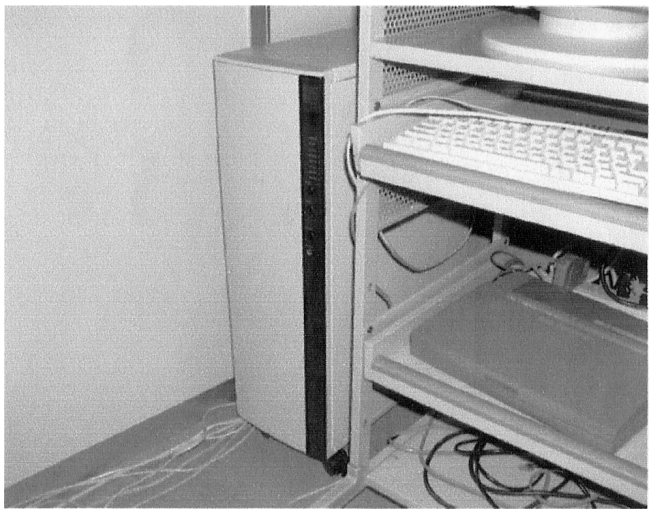

図 5　ファイル・サアバー

この事は、AT 互換機と Alpha CPU 機の性能比較表を見ても分かるようにキャッシュ・メモリーを Alpha CPU では 3 次キャッシュ・メモリーを持っている。したがって、AT 互換機ではキャッシュ・ミスを起こしている可能性がある。(図 4。)

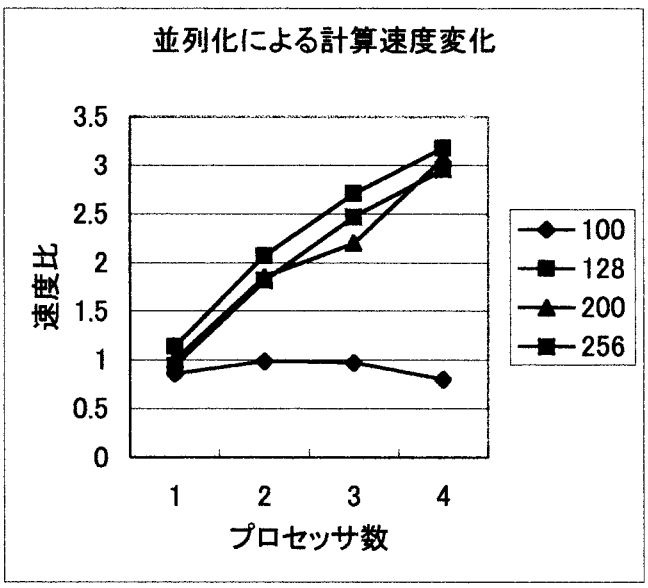

図 6　Alpha CPU を用いた計算速度

7. ZPL による計算結果

FORTRAN や C 言語を用いた並列化では随所に MPI メッセージを挿入する必要がある。しかし、最初から並列化を目指して開発された新しい言語である ZPL では MPI メッセージは一切不要である。恐

らくZPLのコンパイラーの中にMPIメッセージが自動的に挿入されるようになっているものと思う。QR法をZPL言語で書いたリストを参考のために示す。

```
program mat;
config var n : integer = 5;
region R = [1..n,1..n];
region Col = [1..n,*];
region Row = [*,1..n];
region Sca = [1,*];
procedure mat();
var A,Qi,Q,QQQ,Qt,Iden,Dum : [R] double;
        ai,wi,dummy : [Col] double;
        wj : [Row] double;
        i,j,k : integer;
        sigma,sign,alpha : [Sca] double;
        alp : double;
        fp : file;
begin
        [R]Iden := Index1=Index2;
        [R]Qi := Index1=Index2;
        [R]Q := Index1=Index2;
        fp := open("input","r");
        [R]read(fp,A);
        [R]writeln("A=¥n",A);
        close(fp);
        for i:=1 to n-1 do
          [Col] ai:=0;
          [Col] wi:=0;
          [i..n,*] ai:= >>[i..n,i]A;
          [i..n,*] wi:= >>[i..n,i]A;

          [Sca]sigma := +<<[i..n,*](ai*ai);
        [Sca]sign := >>[i,*]((fabs(ai)=ai)-0.5)*2;
[i,*]wi:=>>[i,*]ai+(>>[1,*]sign)*sqrt(>>[1,*]sigma)
  ;
      [Sca]alpha:= +<<[i..n,*](wi*wi);
      [Sca]alpha := 2.0/alpha;
      [Row]wj:= <##[Index2,Index1]wi;
          [R]Qi := wi*wj;
          [1,*]alp := max<<alpha;
          [R]Qi := Iden - alp*Qi;
          [R]Dum := 0;
      for j:=1 to n do
        [Col]wi := >>[1..n,j]Qi;
        [Row]wj := >>[j,1..n]Q;
        [R]Dum := Dum + wi*wj;
      end;
          [R]Q := Dum;
          [R]Dum :=0;
      for j:=1 to n do
          [Col]wi := >>[1..n,j]Qi;
          [Row]wj := >>[j,1..n]A;
          [R]Dum := Dum + wi*wj;
      end;
          [R]A := Dum;
       end;
       [R]Qt := <##[Index2,Index1]Q;
       [R]write("Q=¥n",Qt);
       [R]write("A=¥n",A);
       [1..n,1..n] for k:=1 to n do
          QQQ += (>>[,k] Qt)*(>>[k,] Q);
       end;
       [R] write("I=¥n",QQQ);
end;
```

表2 ZPL言語によるQR法のリスト

このリストを見て、Pascal言語に非常に近いものである。特徴的なものに大型のマトリックスが一行で処理できることである。しかも、並列化が自動的に行われている。

図7にはZPL言語を用いて、相対的な時間を測定した結果を示す。プロセッサ数に反比例して時間が減少しているのが分かる。この言語は米国ではワシントン大学やテキサス大学で研究されており、OSも現在のところフリーで配布されている。まだ、現在は研究段階であるが将来はFORTRANやC言語に代わって並列計算用言語として発展の可能性を持っていると思う。

われわれもこのような新しい言語に挑戦する意欲

がほしいものである。

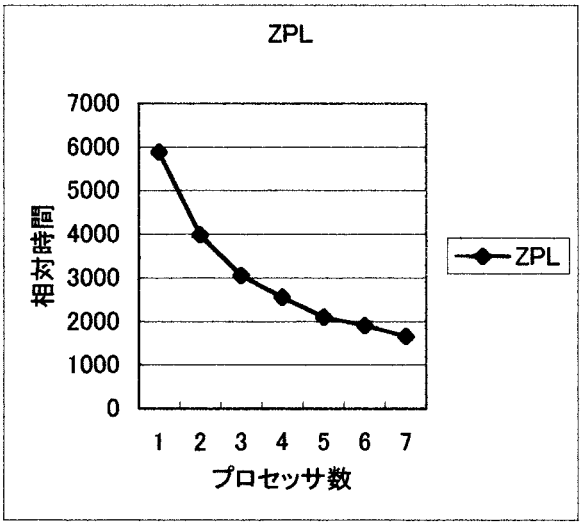

図7 ZPL言語で並列化による相対時間

8. 考察

高速化のボトルネックにTCP/IPが考えられる。この対策に光通信のギガビット・カードが一つ候補に選ばれる。しかし、あまり効果は無かった。

製作した並列計算機はスイッチング・ハブで結ばれた、言わば、クラスターが直線状に結合された計算機である。このネットワークを二次元的に結合したマトリックス状の並列計算機の方がさらに高性能である。また、このクラスターをキュウビック状に結合した方式が考えられるが、まさに、超並列計算機の出現である。

9. おわりに

ノイマン型コンピュータが生まれて、半世紀の間にコンピュータは目覚しく発展し、現在では超並列計算機が稼動している。しかし、この計算機といえども昆虫の脳と比較すれば大変な見劣りのする代物である。つまり、神様には未だ程遠いい存在である。したがって、さらにコンピュータは発展する可能性を残しているのである。せめて、昆虫の脳の性能ぐらいに到達したいものである。

10. 参考文献

1) Message Passing Interface Forum: MPI: A Message-Passing Interface Standard, June 12, 1995

2) William Gropp and Ewing Lusk: Installation Guide to mpich, a Portable Implementation of MPI, ANL/MCS-TM-ANL-96/5

3) William Gropp and Ewing Lusk: User's Guide for mpich, a Portable Implementation of MPI, ANL/MCS-TM-ANL-96/6

4) Peter S. Pacheco and Woo Char Ming: MPI User Guide in FORTRAN, March 17, 1997

5) Calvin Lin: ZPL Language Reference Manual, October 17, 1996

6) 湯浅太一他、初めての並列プログラミング、共立出版 (1998)

7) 和田吉博、並列処理コンピュータを用いた有限要素法による流れ解析、平成8年度卒業論文

8) 浅野裕、並列処理系の構築と運用、平成10年度柳澤研究室卒業論文

9) 折田尚久、並列計算機の構築、平成11年度柳澤研究室卒業論文

10) 近藤威、郡嶋幸夫、MPIを用いた数値計算の並列化、平成11年度柳澤研究室卒業論文