第2回 クラスタゼミ

ゼミ担当者　：　狩野浩一，市川親司，江上透
指導院生　　：　下坂久司，釘井睦和，輪湖純也
開催日　　：　2003 年 5 月 12 日

ゼミ内容: 今回のゼミでは，前回学んだクラスタを実際に用いることで理解を深めることを目的としている．そこで Evolve System を用いて，π 計算を行う．また，ベンチマークについても学ぶ．

1 Evolve System

Evolve System とは，本研究室にある PC クラスタである．PC クラスタとは，複数の PC をネットワークで接続した並列計算機である．

Evolve System の構成については，Fig. 1 のように，evolve という 1 台の PC が forte，sequence，fraulein，mill，g-cherry というクラスタを制御している．各クラスタにはそれぞれマスタノードとスレーブノードが存在する．forte はマスタノード (forte) と 32 台のスレーブノード (forte01〜forte32) の計 33 台，sequence はマスタノード (sequence) と 8 台のスレーブノード (reef-101〜reef-108) の計 9 台，fraulein はマスタノード (fraulein) と 7 台のスレーブノード (freund01〜freund06) の計 7 台で構成されている．

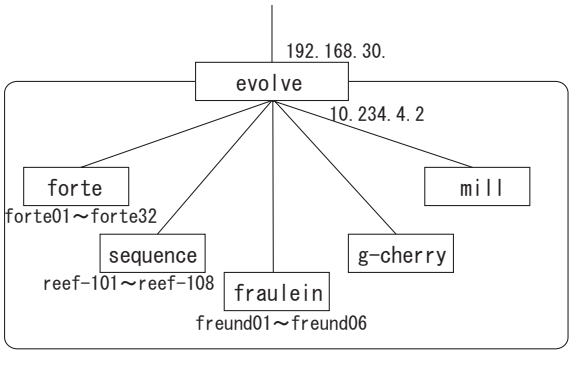

Fig. 1　Evolve System の構成

2 並列計算

MPI を用いたプログラムを使用して，並列計算を行う．PC クラスタ Evolve System を用いて行う．

2.1 π計算のアルゴリズム

並列化を行う対象は π を近似的に求めるプログラムを用いる．π は式 (1) に示す計算式で求めることができる．また，これを逐次的に計算するプログラムは，積分計算が Fig. 2 に示すように行われるので，0 から loop-1 個の区間の積分計算に置き換えられる．この積分計算のループ部分を並列化する．プログラムでは，複数のプロセッサで loop 個の区間の計算を分担するようなアルゴリズムを採用する．(プログラムは後ろに載せている)

$$\pi = \int_0^1 \frac{4}{1+x^2} dx \tag{1}$$

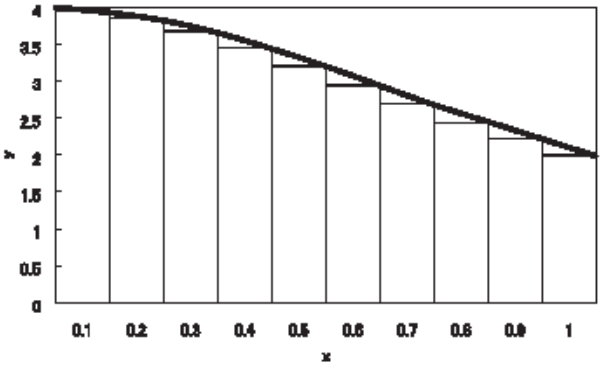

Fig. 2　$y = \frac{4}{1+x^2}$

3 MPICH

MPICH では，ファイルの実行時に様々なオプションをつけてやることで，その動作を制御できる．その方法は以下のようになる．

mpirun（オプションを入れる）（実行プログラム）

3.1 -np

オプションとして -np（数値）を入れることで，プログラムの実行に使用するプロセスの数を設定できる．記入例は以下に示す．

3.2 -machinefile

MPI プログラムでは，machinefile をどのように設定するかによって，並列プロセスをどのプロセッサに割り当てるかを決めることができる．このとき，MPICH に

おける machinefile では，1 つのノード名が複数回現れた場合，対応する複数のプロセスを同一ノードに割り当てる．

```
mpirun -np 3 -machinefile name.LINUX a.out
```

このコマンドは，プログラム a.out を実行する際に使用される．name.LINUX は使用するノードを記述するファイルであり，任意のファイル名をつけられる．例えば，forte01，forte02，forte03 という子ノードを利用したい場合，以下のようなファイルを用意する．

```
forte01
forte02
forte03
```

3.3 -gdb

gdb はデバッガを行うオプションであり，プログラムを実行しながら，その中で一体何が起こっているのかを調査できる．その機能は細かく分けて 4 つである．

- プログラムを起動し，その実行結果が及ぼすであろう様々な事柄を報告する．
- 指定された状況において，プログラムの実行を停止する．(エラーなど)
- 何が起こっているのかを検査する．そのことによって，プログラムが停止した時，バグの原因を知ることができる．
- プログラムの状態を変更することができる．

実際に gdb によるデバッグを行う場合は，オプションとして -gdb を加えてやればよい．その記入例を以下に示す．

```
mpirun -np 3 -gdb -machinefile name.LINUX a.out
```

3.4 -nolocal

nolocal は，ローカルマシンをノードとして動かさないようにするためのオプションである．つまり，各ノードにプログラムを分配するマシンでのプログラムの実行を禁止している．その記入例を以下に示す．

```
mpirun -np 3 -nolocal -machinefile name.LINUX a.out
```

4 LAM

LAM では，ファイルの実行時に様々なオプションをつけてやることで，その動作を制御できる．その方法は以下のようになる．

```
mpirun（オプションを入れる）（実行プログラム）
```

4.1 -O

複数のマシンが均質である場合，メッセージ送信時にデータ変換をしないことで高速化を実現する．その記入例を以下に示す．

```
mpirun -np 3 -O a.out
```

4.2 -toff

プログラム中で MPI_LTrace_on を指示したときのみ trace generation[1]を実行するため，処理速度を高速化できる．その記入例を以下に示す．

```
mpirun -np 3 -toff a.out
```

5 実行手順

MPICH と LAM の実行プログラムが Evolve System では利用できる．両方の実行手順を示していく．

5.1 MPICH の場合

以下のようにコマンドを入力し，PATH を通す．

```
export PATH=/usr/lib/mpich/bin:$PATH
```

次に，以下のようにコマンドを入力し，コンパイルする．

```
mpicc プログラム名.c -o 実行ファイル名
```

利用するマシンを設定するため，以下のように vi や emacs で，利用するマシン名を一つずつ縦に入力し，名前を付けて拡張子「LINUX」として保存する．

```
forte01
forte02
  :
```

ホームの下のディレクトリに，同様に利用するマシン名を書き，名前を付けて拡張子「.rhosts」として保存する．

[1]長時間アプリケーションを動作させたときに発生する不要なデータ群がたまるのを防ぐために，断続的にまとまった演算を送りつづける動作

```
forte01
forte02
   :
```

並列計算を実行する．

```
mpirun -np プロセス数 -machinefile 設定した名前.LINUX 実行ファイル名
```

5.2 LAM の場合

ホームの下のディレクトリのファイル「.bashrc」に次の PATH を追加する．vi や emacs で「.bashrc」を開いて，次の PATH を追加し保存する．

```
export PATH=/usr/lib/lam/bin:$PATH
```

利用するマシンを設定するため，以下のように vi や emacs で，利用するマシン名を一つずつ縦に入力し，名前を付けて拡張子「LINUX」として保存する．

```
forte01
forte02
   :
```

ホームの下のディレクトリに，同様に利用するマシン名を書き，名前を付けて拡張子「.rhosts」として保存する．

```
forte01
forte02
   :
```

正常に動作するのか確認する．

```
recon -v 設定した名前.LINUX
```

デーモンを立ち上げる．

```
lamboot -v 設定した名前.LINUX
```

並列計算を実行する．

```
mpirun -np プロセス数 -machinefile 設定した名前.LINUX 実行ファイル名
```

デーモンを消しておく．

```
wipe -v ホストファイル
```

```c
#include <stdio.h>
int main(int argc, char **argv)
{
    int i,loop;
    double width,x,pai=0.0;
    loop = atoi(argv[1]);
    width = 1.0 / loop;
    for(i=0;i<loop;i++){
        x = (i + 0.5) * width;
        pai += 4.0 / (1.0 +x *x);
    }
    pai = pai / loop;
    printf("PAI = %f¥n",pai);
    return 0;
}
```

Fig. 3　逐次プログラム

6　putty でのログイン方法

まず puttyjp.exe を起動する．

Fig. 5 の画面になるので，ホスト名に「192.168.30.147」を入力し，プロトコルで SSH にチェックを入れる．

Fig. 5

次に，左のカテゴリから SSH を選択すると Fig. 6 の画面になるので，プロトコルオプションで [2only] にチェックを入れる．

次に，左のカテゴリから SSH の認証を選択すると Fig. 7 の画面になるので，認証のためのプライベートキーファイルで [参照] をクリックし，putty のフォルダに保存している「id_dsa.PPK」を選択する．開くをクリックし，自分のアカウント名，パスフレーズを入力すればログインできる．その結果が Fig. 8 である．

```
#include "mpi.h"
#include <stdio.h>
#include <math.h>
double f( double a ){ return (4.0 / (1.0 + a * a)); }
int main( int argc, char *argv[])
{
    int done = 0, n, myid, numprocs, i;
    double PI25DT = 3.14159265358979323846264 3;
    double mypi, pi, h, sum, x;
    double startwtime, endwtime;
    int namelen;
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,&myid);
    MPI_Get_processor_name(processor_name,&namelen);
    fprintf(stderr,"Process %d on %s¥n",myid, processor_name);
    n =0;
    while (!done){
        if (myid == 0){
            /*
            printf("Enter the number of intervals: (0 quits) ");
            scanf("%d",&n);
            */
            if (n==0) n=100; else n=0;
                startwtime = MPI_Wtime();
            }
        MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
        if (n == 0)
            done = 1;
        else{
                h = 1.0 / (double) n;
                sum = 0.0;
                for (i = myid + 1; i <= n; i += numprocs){
                    x = h * ((double)i - 0.5);
                    sum += f(x);
                }
            mypi = h * sum;
            MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0,
    MPI_COMM_WORLD);
                if (myid == 0){
                    printf("pi is approximately %.16f, Error is %.16f¥n",
    pi, fabs(pi - PI25DT));
                    endwtime = MPI_Wtime();
                    printf("wall clock time = %f¥n",endwtime-startwtime);
                }
        }
        MPI_Finalize();
        return 0;
    }
}
```

Fig. 4　並列プログラム

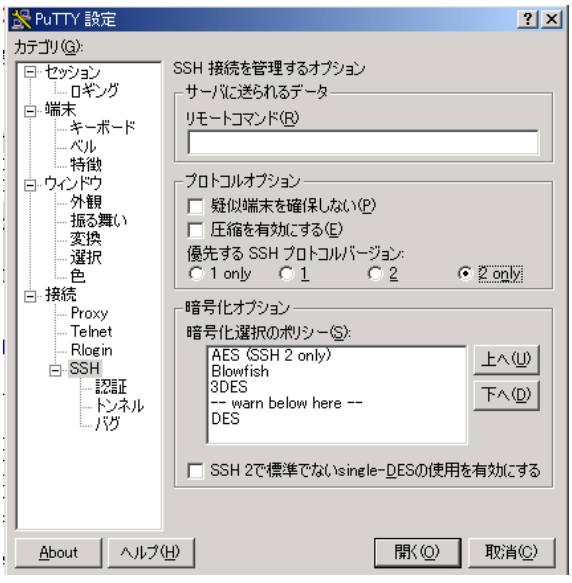

Fig. 6

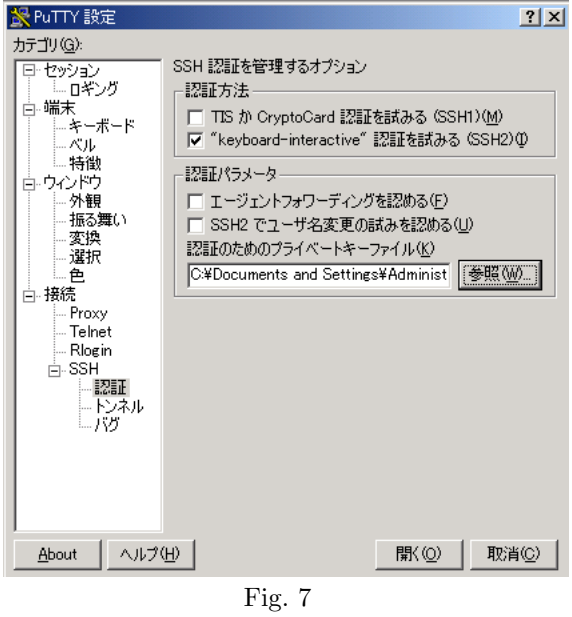

Fig. 7

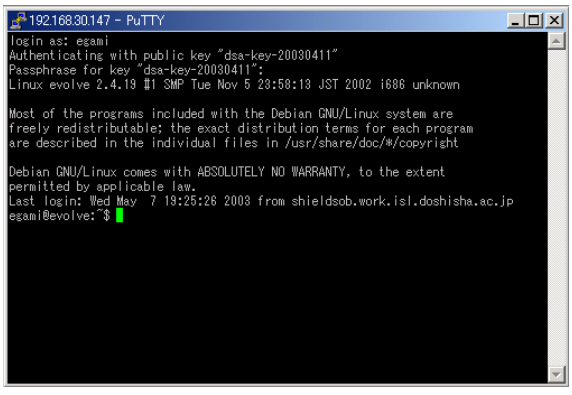

Fig. 8

7 ベンチマークとは

ベンチマークという言葉には,「基準となるもの」という意味がある．コンピュータ用語としてのベンチマークの意味は，ハードウェアやソフトウェアの処理速度を計測する試験問題，およびそれに性能評価という意味を持つ．つまり，計算機システムの性能を計測することを意味する．

コンピュータプログラムによって，さまざまな挙動を示すため，各用途に応じた評価の指標を決定し，その尺度にもとづいて性能を評価することである．ベンチマークの対象となる機能を分類すると，以下のようになる．

- 演算性能ベンチマーク
- I/O 性能ベンチマーク
- グラフィックス関連性能ベンチマーク
- ネットワーク関連性能ベンチマーク
- データベース性能ベンチマーク
- アプリケーション指向ベンチマーク
- 並列/大規模計算ベンチマーク

PC クラスタに関係する並列計算ベンチマークとして挙げられるのが，Nas Parallel Benchmark，ScaLAPACK，姫野ベンチ，GA Benchmark である．今回使用するのは，姫野ベンチマークで，以下の部分に姫野ベンチの説明をする．

8 姫野ベンチとは

姫野ベンチマークは理化学研究所情報環境室・室長の姫野龍太郎氏が非圧縮流体解析コードの性能評価のために考えたもので，ポアッソン方程式解法をヤコビの反復法で解く場合に主要なループの処理速度を計るものである．

姫野ベンチマークテストは主に計算機のメモリのバンド幅などの性能による効果が大きいベンチマークである．コードは非常に短く簡単にコンパイル．実行ができ，約 1 分間で実測速度（何 MFLOPS）を求めることができる．

9 姫野ベンチの実行方法

姫野ベンチには多くの種類が存在するが，今回は Fortran + MPI 版と 1CPU 版についての実行方法についてみることにする．

9.1 Fortran + MPI 版姫野ベンチ実行方法

まず，http://w3cic.riken.go.jp/HPC/HimenoBMT/Load_module/f77_xp_mpi.lzh からソースをダウンロードする．

次にダウンロードしたファイルの解凍を行う．

```
lha -x f77_xp_mpi.lzh
```

すると，以下のファイルが作成される．

- himenoBMTxpr.f ··· 実行ファイル
- param.h ··· パラメータファイル
- paramset.sh ··· パラメータを作るスクリプト

次にパラメータを設定を以下の要領で行う．

```
chmod 775 paramset.sh
```

```
./paramset.sh [計算サイズ a] [分割数 x] [分割数 y]
[分割数 z]
```

すると新しい param.h というファイルが作成される．計算サイズには，XS，S，M，L，XL がある．ここで分割数 x，y，z を全てかけると，使用する CPU 数になるようにしなければならない．

そして，実行ファイルである himenoBMTxpr.f のコンパイルを行う．

```
mpif77 himenoBMTxpr.f
```

そうすれば実行可能となる．

以下に実行例を示す．

```
mpirun -np 16 a.out
```

9.2 1CPU 版姫野ベンチの実行方法

まず，http://w3cic.riken.go.jp/HPC/HimenoBMT/Load_module/himenoBMTxp_s.lzh からソースファイルをダウンロードする．

次にダウンロードしたファイルの解凍を行う．

```
lha -x himenoBMTxp_s.lzh
```

すると himenoBMTxp_s.f というソースファイルが生成される．

次に UNIX の場合，コンパイルを行う前に himenoBMTxp_s.f の 42 行目をコメントアウト (もしくは削除) しておかなければならない．

```
C ”use portlib” statement on the next line is
C to use UNIX libraries. Please remove it if
C ————————M
use portlibM (ここをコメントアウトする)
IMPLICIT REAL*4(a-h,o-z)M
```

そして，コンパイルを行う．

```
g77 himenoBMTxp_s.f
```

あとは実行するだけである．

```
./a.out
```

10 姫野ベンチ Windows･Mac での実行方法

実際に，自分のコンピュータの計算速度を計るために，http://w3cic.riken.go.jp/HPC/HimenoBMT/program1.htm でダウンロードする．その Web 上で実行形式である，「s サイズ」をダウンロードする．その図を Fig. 9 に示す．

実行形式

サイズ	L (512 x 256 x 256)	M (256 x 128 x 128)	S (128 x 64 x 64)	XS
実行形式 (Win 版)	himenoBMTxp_l.exe	himenoBMTxp_m.exe	himenoBMTxp_s.exe	
	圧縮ファイル (LZH)	圧縮ファイル (LZH)	圧縮ファイル (LZH)	
実行形式 (Mac 版)	-	himenoBMT98xp_m_mac	himenoBMT98xp_s_mac	himenoBMT98xp_ss_mac
	-	圧縮ファイル (LZH)	圧縮ファイル (LZH)	圧縮ファイル (LZH)

Fig. 9 姫野ベンチ・ダウンロード

ファイルをダウンロードして，解凍すると himenoBMTxp_s.exe という exe ファイルができるので，それをクリックして 1 分間待つと，自分のコンピュータの計算速度が計ることができる．以下のような計算結果が出る（Fig. 10）．

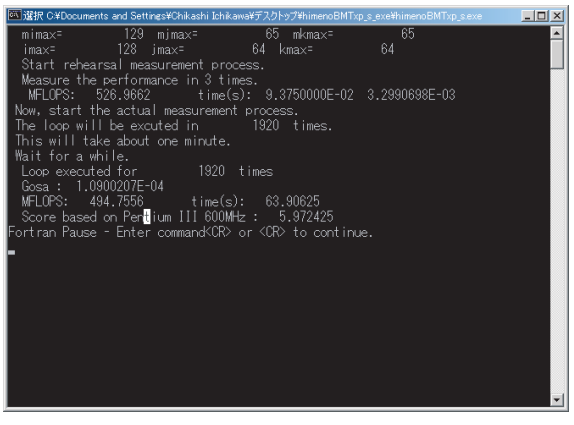

Fig. 10 姫野ベンチ・計算結果