スーパーコンピュータの基本的操作方法

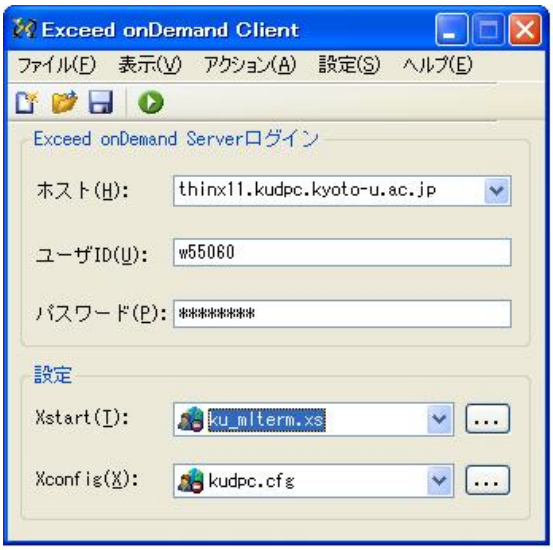
2009 年 9 月 10 日

高橋康人

1. スーパーコンピュータへのログイン方法

※本演習では，X端末ソフト Exceed on Demand を使用するが，必要に応じて SSH クライアント putty，FTP クライアント WinSCP や FileZilla を使用して構わない。

Exceed on Demand を起動し，以下のとおり設定（各自のユーザ ID とパスワード*を入力）した上で緑の矢印のボタンでログインする。詳細については，http://web.kudpc.kyoto-u.ac.jp/hpc/eod を参照せよ。



※スーパーコンピュータ用のパスワードを設定できていない場合，
https://web.kudpc.kyoto-u.ac.jp/portal/ecs/pwinit
からパスワードを設定する。

机の上のマニュアルには Exceed on Demand と FileZilla の使用法が書かれているので，必要に応じて参照せよ。また，putty，WinSCP の使用方法については，以下のホームページを参考にせよ。
http://web.kudpc.kyoto-u.ac.jp/hpc/node/339

2 コンパイルの方法と会話型の実行方法

2.1 逐次実行の場合

2.1.1 コンパイル

(1) C 言語（詳細は，http://web.kudpc.kyoto-u.ac.jp/hpc/fujitsu/c）

fcc [オプションの並び] [ファイル名の並び] （例．fcc test.c ← test.c を翻訳，結合編集）

(2) Fortran（詳細は，http://web.kudpc.kyoto-u.ac.jp/hpc/fujitsu/fortran）

frt [オプションの並び] [ファイル名の並び] （例．frt test.f90 ← test.f90 を翻訳，結合編集）

2.1.2　実行方法

./ [実行ファイル名]　　（例．./a.out ← 実行）

また，実行時間を知りたい場合には，time コマンドを使って下記のように実行する。

time ./a.out ← 実行時間の計測

・tcsh の場合

6.394u 0.091s 0:06.59 98.3%　　0+0k 0+0io 1pf+0w ← 出力結果

左から順に，ユーザー時間（計算時間）：6.394 秒，システム時間：0.091 秒，実時間（コマンドを起動してから終了するまでの経過時間）：0 分 6.59 秒を表す。実時間を見ればよい。

・bash の場合

real　0m6.425s　（実時間（経過時間））

user　0m6.252s　（ユーザー時間（計算時間））

sys　0m0.040s　（システム時間）

2.1.3 主要なコンパイルオプション

-c：オブジェクトファイルの作成までを実行。

-o name：実行可能ファイル，オブジェクトファイルの名前を name に変更（省略時は a.out）

-Kfast：高速化のための最適化オプション。

2.2　OpenMP のディレクティブを有効化するコンパイルと実行方法

（詳細：http://web.kudpc.kyoto-u.ac.jp/hpc/node/141）

2.2.1　コンパイルオプション

翻訳・結合編集には，コンパイルコマンド（frt, fcc）に「-KOMP」オプションを指定。

例．（C 言語）fcc -KOMP test_omp.c

（Fortran）frt -KOMP test_omp.f90

2.2.2　スレッド数の指定・実行

・tcsh の場合

$ echo $OMP_NUM_THREADS ← 現在の環境変数 OMP_NUM_THREADS の値を確認

4 ← 現在の環境変数 OMP_NUM_THREADS には 4 が設定されている

$ ./a.out ← 実行 (4 並列)

$ setenv OMP_NUM_THREADS 8 ← 並列数を 8 に指定

$ ./a.out ← 実行 (8 並列)

・bash の場合

$ echo $OMP_NUM_THREADS ← 現在の環境変数 OMP_NUM_THREADS の値を確認

4 ← 現在の環境変数 OMP_NUM_THREADS には 4 が設定されている

$ ./a.out ← 実行 (4 並列)

$ export OMP_NUM_THREADS=8 ← 並列数を 8 に指定

2.3　MPI プログラムのコンパイルと実行方法（詳細：http://web.kudpc.kyoto-u.ac.jp/hpc/node/143）

2.3.1　コンパイル

(1) C 言語・・・mpifcc　[オプションの並び]　[ファイル名の並び]　（例．mpifcc test.c）

(2) Fortran・・・mpifrt　[オプションの並び]　[ファイル名の並び]　（例．mpifrt test.f90）

2.3.2　実行方法

mpiexec　-n N　実行ファイル名　　（-n N：プロセスの並列数 N を指定）

例．$ mpiexec -n 8 ./a.out　（8 並列で実行）

3. NQS の使用方法（http://web.kudpc.kyoto-u.ac.jp/hpc/nqs）

3.1　ジョブの実行方法（スクリプトを実行ファイルがあるディレクトリに移動しておく。）

・ジョブの投入：　　$ qsub sample.sh

・ジョブの確認：　　$ qstat

・ジョブの途中経過確認：$ qcat 12345.nqs

・ジョブのキャンセル：　$ qdel 12345.nqs　（ジョブをキャンセル）

　　$ qdel -k 12345.nqs　（実行中のジョブをキャンセル）

・結果ファイル：　　Nxxxxxx.oxxxxxx

例．$ qsub nqs.sh -lp 8　（スレッド数をコマンドラインで指定）

　　$ qsub nqs.sh -lP 8　（プロセス数をコマンドラインで指定）

　　$ qsub nqs.sh -lm 1.2gb　（プロセスあたりのメモリサイズの指定）

3.2　ジョブスクリプトの形式

```
# この例は，MPI で 2 並列，スレッド並列で 4 並列 -> 計 2x8
#
# @$-eo          # 標準出力標準エラー出力をまとめる
# @$-Ol          # 統計情報を出力する（-oi・・・ノード情報も含めて統計情報の標準出力）
# @$-q eh        # キュー名の指定　（各自設定：sh20103（情報学研究科）or qh10160（それ以外））
# - eh はエントリコース用のキュー
# グループコース用のキュー名はグループ名と同じ名前。
# 例えば ghXXXXX のグループの場合，キュー名は ghXXXXX。
# @$-g eh        # グループの指定。
# キューの指定で指定したキュー名と同じ名前をグループ名として指定。

# @$-lP 2        # プロセス数　　（各自設定）
# @$-lp 4        # プロセスあたりの CPU 数（スレッド並列数）　　（各自設定）
# @$-lm 1gb      # プロセスあたりのメモリサイズ　　（各自設定）
set -x

# qsub コマンド実行時のディレクトリに移動
cd $QSUB_WORKDIR

# 環境変数 OMP_NUM_THREADS の定義
OMP_NUM_THREADS=$QSUB_CPUS; export OMP_NUM_THREADS

# MPI プログラム ./a.out を実行
mpiexec -n $QSUB_VNODES ./a.out　　（各自設定）
```

“#” 以降はコメント。“#” に続けて “@$” と記述した場合は，それ以降が NQS のオプション。
サンプルスクリプトを参考に，おもに下線部を変更すればよい。
サンプルスクリプトが以下の URL から入手できるので，必要に応じて参考にせよ。
http://web.kudpc.kyoto-u.ac.jp/hpc/nqs/sample

4. サンプルプログラムの実行

4.1 逐次実行

・サンプルプログラムを作成（C または Fortran のどちらか一方で行えばよい）

（C 言語）

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <sys/time.h>
#include <sys/resource.h>
#define N 1024
#define ZERO (double)(0.0)
#define THREE (double)(3.0)

double getrusage_sec(){
  struct rusage t;
  struct timeval tv;
  getrusage(RUSAGE_SELF, &t);
  tv = t.ru_utime;
  return tv.tv_sec + (double)tv.tv_usec*1e-6;
}

int main(){
  static int i,j,k;
  static double a[N][N],b[N][N],c[N][N],s;
  static double t1,t2;

  srand(1);
  for(i=0;i<N;i++){
    for(j=0;j<N;j++){
      a[i][j]=rand()/(double)RAND_MAX;
      b[i][j]=rand()/(double)RAND_MAX;
    }
  }
  t1 = getrusage_sec();
  for(i=0;i<N;i++){
    for(j=0;j<N;j++){
      c[i][j]=ZERO;
      for(k=0;k<N;k++){
        c[i][j]=c[i][j]+a[i][k]*b[k][j]/THREE;
      }
    }
  }
  t2 = getrusage_sec();
  printf("time = %10.5f¥n", t2 - t1);
  s=ZERO;
  for(i=0;i<N;i+=10){
    for(j=0;j<N;j+=10){
      if(a[i][j]>s) s=a[i][j];
      if(b[i][j]>s) s=b[i][j];
      if(c[i][j]>s) s=c[i][j];
    }
  }
  printf("%f¥n",s);
  return 0;
}
```

（Fortran）

```
program main
  integer i, j, k
  parameter (N=1024, ZERO=0.0d0, THREE=3.0d0)
  double precision A(N,N),B(N,N),C(N,N),s
  real t1,t2

  call random_number(A)
  call random_number(B)

  call cpu_time(t1)
  do i=1,N
     do j=1,N
        C(i,j)=ZERO
        do k=1,N
           C(i,j)=C(i,j)+A(i,k)*B(k,j)/THREE
        end do
     end do
  end do
  call cpu_time(t2)

  write(6,*) (t2-t1), ' sec (CPU time)'

  s=ZERO
  DO I=1,N,10
     DO J=1,N,10
        if(A(J,I) .gt. s) then
           S=A(j,i)
        end if
        if(B(j,i) .gt. s) then
           S=B(j,i)
        end if
        if(C(j,i) .gt. s) then
           S=C(j,i)
        end if
     end do
  end do
  write(6,*) s

end program main
```

・コンパイル

fcc –Kfast –o sample1 single_test.c　　または　　frt –Kfast –o sample1 single_test.f90

・実行

./sample1

・NQS ジョブの投入・・・上記 URL の nqs_group_single.sh を参考にして，スクリプトを編集。

4.2　OpenMP を用いた並列計算

・サンプルプログラムを作成（C または Fortran のどちらか一方で行えばよい）

（C 言語）

```
#include "omp.h"
#include <stdio.h>
#include <math.h>
#include <sys/time.h>

int main(){
  int split, i ;
  double height, width, area, pai, time ;
  struct timeval stp, etp ;

  gettimeofday(&stp, NULL) ;

  area = 0 ;
  split = 20000000 ;
  width = 1.0 / (double)split ;

  #pragma omp parallel for reduction(+:area) private(height)
  for(i=0; i<split; i++){
    height = sqrt(1.0 - pow((i * width), 2.0)) ;
    area = area + height * width ;
  }

  pai = 4.0 * area ;
  printf("pai = %lf¥n", pai) ;

  gettimeofday(&etp, NULL) ;
  time = (double)(etp.tv_usec - stp.tv_usec) / 1000000.0 ;
  time += (double)(etp.tv_sec - stp.tv_sec) ;
  printf("%lf sec¥n", time) ;

return 0;
}
```

（Fortran）

```
      program circle
      integer split, i
      real*8 height, width, area, pai, stime, etime

      call gettod(stime)
      area = 0
      split = 20000000
      width = 1.d0 / split
!$OMP PARALLEL DO REDUCTION(+ : area) PRIVATE(height)
      do i = 0, split-1
          height = sqrt(1.d0 - (real(i) * width) ** 2)
          area = area + height * width
      enddo
!$OMP END PARALLEL DO

      pai = 4.d0 * area
      print *, "pai = ", pai
      call gettod(etime)
      print *, (etime - stime) / 1000000.0 , "sec"
      end
```

・コンパイル

fcc –KOMP –o sample2 omp_test.c　　または　　frt –KOMP –o sample2 omp_test.f90

・スレッド数を変更して実行

./sample2

setenv OMP_NUM_THREADS 8

./sample2

・NQS ジョブの投入・・・上記 URL の　nqs_group_thread.sh を参考にして，スクリプトを編集。

4.3　MPI を用いた並列計算

・サンプルプログラムを作成（C または Fortran のどちらか一方で行えばよい）

（C 言語）

```
#include <stdio.h>
#include "mpi.h"
#define PROCS 4

int main(int argc, char *argv[]){
  double a, b;
  int size_world,rank_world;

  MPI_Init(&argc,&argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank_world);
  MPI_Comm_size(MPI_COMM_WORLD, &size_world);

  if(size_world != PROCS){
    printf("Program needs %d processes.¥n", PROCS);
    MPI_Abort(MPI_COMM_WORLD, 1);
    return (1);
  }

  if ( rank_world==0 ){
    a = 3.0;    b = 2.0;
    printf("a=%lf, b=%lf¥n",a,b);
  }

  MPI_Bcast(&a,1,MPI_DOUBLE,0,MPI_COMM_WORLD);
  MPI_Bcast(&b,1,MPI_DOUBLE,0,MPI_COMM_WORLD);

  switch( rank_world ){
  case 0:
    printf("rank=%d a+b=%lf¥n",rank_world,a+b);
  break;
  case 1:
    printf("rank=%d a-b=%lf¥n",rank_world,a-b);
  break;
  case 2:
    printf("rank=%d a*b=%lf¥n",rank_world,a*b);
  break;
  case 3:
    printf("rank=%d a/b=%lf¥n",rank_world,a/b);
  break;
  }

  MPI_Finalize();
  return 0;
}
```

（Fortran）

```
program mpi_test
  include 'mpif.h'
  double precision a, b
  integer rank_world, size_world, ierror
  PARAMETER (PROCS=4)

  call MPI_Init(ierror)
  call MPI_Comm_rank(MPI_COMM_WORLD, rank_world, ierror)
  call MPI_Comm_size(MPI_COMM_WORLD, size_world, ierror)

  if( size_world .ne. PROCS ) then
    write(6,*) 'Program needs', PROCS, ' processes.'
    call MPI_Abort(MPI_COMM_WORLD, 9, ierror)
  end if

  if ( rank_world .eq. 0 ) then
     a=3.d0
     b=2.d0
     write(6,*) 'a=', a, ' b=', b
  end if

  call MPI_Bcast(a,1,MPI_DOUBLE,0,MPI_COMM_WORLD,ierror)
  call MPI_Bcast(b,1,MPI_DOUBLE,0,MPI_COMM_WORLD,ierror)

  select case(rank_world)
  case(0)
     write(6,*) 'rank=', rank_world, ' a+b=', a+b
  case(1)
     write(6,*) 'rank=', rank_world, ' a-b=', a-b
  case(2)
     write(6,*) 'rank=', rank_world, ' a*b=', a*b
  case(3)
     write(6,*) 'rank=', rank_world, ' a/b=', a/b
  end select

  call MPI_Finalize(ierror)
end program mpi_test
```

・コンパイル

mpifcc –o sample3 mpi_test.c　　または　　mpifrt–o sample3 mpi_test.f90

・プロセス数を 4 として実行

mpiexec –n 4 ./sample3

・NQS ジョブの投入・・・上記 URL の　nqs_group_mpi.sh を参考にして，スクリプトを編集。