Prof. Dr. O. Rheinbach
Dipl.-Math. F. Röver
Institut für Numerische Mathematik und Optimierung
TU Bergakademie Freiberg

1st Exercise in HPC

Exercise 1
Use the following program to find out how OpenMP parallelizes the loop. Compile using `gcc -O -fopenmp omp-forloop.c`

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

int main()
{
  int i;
  int N=10;
  int num;
#pragma omp parallel for private(num)
  for(i=0;i<N;i++)
  {
    num=omp_get_thread_num();
    printf("Thread %d does iteration %d\n",num,i);
  }
}
```

Exercise 2
Test the following OpenMP-Programm with respect to parallel performance for different numbers of threads, i.e. use

```
export OMP_NUM_THREADS=12
export OMP_NUM_THREADS=6
export OMP_NUM_THREADS=3
export OMP_NUM_THREADS=2
export OMP_NUM_THREADS=1
```

and run it

```
time ./a.out
```

Also compare it with a sequential version, i.e., a version compiled without OpenMP.

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
```

```
// compile: gcc -O -fopenmp matrixmult.c

void multiply(const int N, const double *a, const double *b, double *c)
{
  int i,j,k;

#pragma omp parallel for private(i,j,k)
  for(i=0;i<N;i++)
  {
    for(j=0;j<N;j++)
    {
      c[i*N+j]=0;
      for(k=0;k<N;k++)
      {
        c[i*N+j]+=a[i*N+k]*b[k*N+j];
      }
    }
  }
}

void print(const double *a, int N)
{
  int i,j;
  for(i=0;i<N;i++)
  {
    for(j=0;j<N;j++)
    {
      printf(" %f",a[i*N+j]);
    }
    printf("\n");
  }
  printf("\n");
}

int main()
{
  const int N=2000;
  //const int N=3;
  int i,j,k;

  double *a=(double*)malloc(sizeof(double)*N*N);
  double *b=(double*)malloc(sizeof(double)*N*N);
  double *c=(double*)malloc(sizeof(double)*N*N);

#pragma omp parallel
  {
    int numthreads,num;
    numthreads=omp_get_num_threads();
    num=omp_get_thread_num();
    printf("Thread %d of %d\n",num,numthreads);
  }
```

```
#pragma omp parallel for private(i,j)
  for (i=0; i<N; i++)
  {
    for (j=0; j<N; j++)
    {
      a[i*N+j]= i+j+1.0;
      b[i*N+j]= 1.0/(i+j+1);
      c[i*N+j]= 0;
    }
  }

  for(i=0;i<1;i++)
  {
    multiply(N,a,b,c);
  }

#pragma omp master
  {
    if(N<20)
    {
      print(a,N);
      print(b,N);
      print(c,N);
    }
    else
    {
      printf("N too big. I do not print.\n");
    }
  }
}
```

Compare the results also to you own implementation of the matrix-matrix-multiplication.

Remark: If you are using your laptop you may encounter segmentation faults in your own implementation for large arrays, i.e., for `double a[1000][1000]`. In this case the default stack size may be too small. Try

```
export GOMP_STACKSIZE="1000000"
ulimit -s unlimited
```

Remarks: How to Compile and Run OpenMP programs

Log on to the login node of the HPC cluster using ssh (Linux)

```
$>  ssh <YOURLOGIN>@login01.hrz.tu-freiberg.de
```

or use an ssh client under Windows (Putty etc.).

You can use any editor to edit your program file (e.g., `vi, emacs`) but one of the simplest to use is `nano`. Use `gcc -fopenmp` to compile C programs and `g++ -fopenmp` to compile C++ programs with OpenMP.

```
$>  nano loop.c
$>  gcc -fopenmp loop.c
```

You can change the number of threads by setting `OMP_NUM_THREADS` before running the program.

```
$>  export OMP_NUM_THREADS=4
$>  time ./a.out
```

You can see all the processes that run on your machine by

```
$>  top
```

You can exit `top` by entering `q`.

By default you are logged into the login node. If you run executables on the login node all users will compete for the processor resources. You can see who is logged in to the login node by typing `finger`.

Instead you can obtain an interactive shell on a separate compute node by asking the queuing system for an interactive job

```
$>  qsub -I
```

Remark: The option is a capital `i` (=interactive) not a number 1. After a short time you will be logged into a separate compute node, e.g., **node110** and moved to a subdirectory,
e.g., `pbs.582279.master.cm.cluster.x8z`.

Move upwards

```
$>  cd ..
```

to your home directory to find your source files and programs.

You can find details on the queueing system on the homepage of the cluster

`http://tu-freiberg.de/urz/dienste/hpc`