Exercise-5

Serial code is available at (`./trainingIPT/exercises/exercises_3_to_7/md.c`):

The complete code for this exercise can also be obtained from:
http://people.sc.fsu.edu/~jburkardt/c_src/md/md.html

Snippet of the serial code:

```
void compute ( int np, int nd, double pos[], double vel[], double
mass,double f[], double *pot, double *kin )
{
  double d;
  double d2;
  int i;
  int j;
  int k;
  double ke;
  double pe;
  double PI2 = 3.141592653589793 / 2.0;
  double rij[3];

  pe = 0.0;
  ke = 0.0;

  for ( k = 0; k < np; k++ )
  {
/*
  Compute the potential energy and forces.
*/
    for ( i = 0; i < nd; i++ )
    {
      f[i+k*nd] = 0.0;
    }

    for ( j = 0; j < np; j++ )
    {
      if ( k != j )
      {
        d = dist ( nd, pos+k*nd, pos+j*nd, rij );
/*
  Attribute half of the potential energy to particle J.
```

```
*/
        if ( d < PI2 )
        {
          d2 = d;
        }
        else
        {
          d2 = PI2;
        }

        pe = pe + 0.5 * pow ( sin ( d2 ), 2 );

        for ( i = 0; i < nd; i++ )
        {
          f[i+k*nd] = f[i+k*nd] - rij[i] * sin ( 2.0 * d2 ) / d;
        }
      }
    }
/*
  Compute the kinetic energy.
*/
    for ( i = 0; i < nd; i++ )
    {
      ke = ke + vel[i+k*nd] * vel[i+k*nd];
    }
  }

  ke = ke * 0.5 * mass;

  *pot = pe;
  *kin = ke;

  return;
}
```

Ritu Arora 9/13/2017 10:26 AM
Comment [1]: hotspot for parallelization

Steps for Using IPT

login3$ idev

c557-202$ source runBeforeIPT.sh

c557-903$ ../../IPT md.c
"/work/01698/rauta/trainingIPT/exercises/exercises_3_to_7/md.c", line 66: warning:
variable "i" was declared but never referenced
int i;
^

"/work/01698/rauta/trainingIPT/exercises/exercises_3_to_7/md.c", line 67: warning:
variable "id" was declared but never referenced
int id;
^

"/work/01698/rauta/trainingIPT/exercises/exercises_3_to_7/md.c", line 636: warning:
variable "len" was set but never used
size_t len;
^

Ritu Arora 9/13/2017 10:31 AM
Comment [2]: Related to the input program

NOTE: We currently support only C and C++ programs.

Please select a parallel programming model from the following available options:
1. MPI
2. OpenMP
3. CUDA
2

NOTE: As per the OpenMP standard, a parallelized region/block of statements can have only one entry point and only one exit point. Branching out or breaking prematurely from a parallelized region/block of statements is not allowed. Please make sure that there are no return/break statements in the region selected for parallelization. However, exit/continue statements are allowed in parallel regions.

A list containing the functions in the input file will be presented, and you may want to select one function at a time to parallelize it using multi-threading.

Please choose the function that you want to parallelize from the list below
1 : main
2 : compute
3 : cpu_time
4 : dist

5 : initialize
6 : r8mat_uniform_ab
7 : timestamp
8 : update
2

Please select one of the following options (enter 1 or 2 or 3)
1. Create a parallel region (a group of threads will be created and each thread will execute a block of code redundantly but in parallel)
2. Parallelize a for-loop (a group of threads will be created and each thread will execute a certain number of iterations of a for-loop)
3. Create a parallel section (TBD - this mode is currently unavailable)
2

Note: With your response, you will be selecting or declining the parallelization of the outermost for-loop in the code region shown below. If instead of the outermost for-loop, there are any inner for-loops in this code region that you are interested in parallelizing, then, you will be able to select those at a later stage.

```
for (k = 0; k < np; k++) {
/*
  Compute the potential energy and forces.
*/
  for (i = 0; i < nd; i++) {
    f[i + (k * nd)] = 0.0;
  }
  for (j = 0; j < np; j++) {
    if (k != j) {
      d = dist(nd,(pos + (k * nd)),(pos + (j * nd)),rij);
/*
  Attribute half of the potential energy to particle J.
*/
      if (d < PI2) {
        d2 = d;
      }
      else {
        d2 = PI2;
      }
      pe = (pe + (0.5 * pow(sin(d2),2)));
      for (i = 0; i < nd; i++) {
        f[i + (k * nd)] = (f[i + (k * nd)] - ((rij[i] * sin((2.0 * d2))) / d));
      }
```

```
    }
  }
/*
  Compute the kinetic energy.
*/
  for (i = 0; i < nd; i++) {
    ke = (ke + (vel[i + (k * nd)] * vel[i + (k * nd)]));
  }
}
```

Is this the for loop you are looking for?(y/n)

y

Reduction variables are the variables that should be updated by the OpenMP threads and then accumulated according to a mathematical operation like sum, multiplication,etc.

Do you want to perform reduction on any variable ?(Y/N)

y

Please select a variable to perform the reduction operation on (format 1,2,3,4 etc.). List of possible variables are:

1. nd type is int
2. k type is int
3. np type is int
4. d type is double
5. PI2 type is double
6. d2 type is double
7. pe type is double
8. ke type is double

7,8

Ritu Arora 9/13/2017 10:30 AM
Comment [3]: Two reduction variables

Please enter the type of reduction you wish for variable [pe]

1. Addition
2. Subtraction
3. Min
4. Max
5. Multiplication

1

Please enter the type of reduction you wish for variable [ke]

1. Addition
2. Subtraction
3. Min
4. Max
5. Multiplication

1

IPT is unable to perform the dependency analysis of the array named [f] in the region of code that you wish to parallelize. Please enter 1 if the entire array is being updated in a single iteration of the loop that you selected for parallelization, or, enter 2 otherwise.
2

IPT is unable to perform the dependency analysis of the array named [rij] in the region of code that you wish to parallelize. Please enter 1 if the entire array is being updated in a single iteration of the loop that you selected for parallelization, or, enter 2 otherwise.
1

Are there any lines of code that you would like to run either using a single thread at a time (hence, one thread after another), or using only one thread?(Y/N)
n

Would you like to parallelize another loop in the previously selected function or another one?(Y/N)
n

Are you writing/printing anything from the parallelized region of the code?(Y/N)
n

Running Consistency Tests

Compiling and Running the Generated Code

```
c557-903$ ls -ltr
total 109
-rw-r--r-- 1 rauta G-25072  1358 Sep 11 17:01 README.txt
-rw-r--r-- 1 rauta G-25072 14726 Sep 11 17:01 md.c
-rw-r--r-- 1 rauta G-25072  1286 Sep 11 17:01 heat_serial.c
-rw-r--r-- 1 rauta G-25072  5555 Sep 11 17:01 circuit.c
-rw-r--r-- 1 rauta G-25072    81 Sep 11 17:01 calc_up.h
-rw-r--r-- 1 rauta G-25072   184 Sep 11 17:01 calc_up.c
-rw-r--r-- 1 rauta G-25072  1637 Sep 12 21:05 rose_heat_serial_OpenMP.c
-rwxr-xr-x 1 rauta G-25072 62658 Sep 12 21:08 rose_heat_serial_OpenMP
-rw-r--r-- 1 rauta G-25072 14313 Sep 12 21:29 rose_md_OpenMP.c
c557-903$ icc -o rose_md_OpenMP rose_md_OpenMP.c
rose_md_OpenMP.c(231): warning #3180: unrecognized OpenMP #pragma
  #pragma omp parallel default(none) shared(pe,ke,np,f,pos,vel,nd,PI2) private(k,i,j,d,d2)
firstprivate(rij)
```

```
        ^

rose_md_OpenMP.c(234): warning #3180: unrecognized OpenMP #pragma
  #pragma omp for reduction ( + :pe,ke)
        ^

c557-903$ icc -qopenmp -o rose_md_OpenMP rose_md_OpenMP.c

c557-903$ export OMP_NUM_THREADS=1
c557-903$ time ./rose_md_OpenMP 2 500 500 0.01
12 September 2017 09:30:14 PM

MD
  C version
  A molecular dynamics program.

  ND, the spatial dimension, is 2
  NP, the number of particles in the simulation, is 500
  STEP_NUM, the number of time steps, is 500
  DT, the size of each time step, is 0.010000

  At each step, we report the potential and kinetic energies.
  The sum of these energies should be a constant.
  As an accuracy check, we also print the relative error
  in the total energy.

      Step      Potential       Kinetic        (P+K-E0)/E0
                Energy P        Energy K       Relative Energy Error

         0   122092.740143        0.000000    0.000000e+00
        50   118856.083102     3379.435004    1.169422e-03
       100   118817.342909     4258.011058    8.048094e-03
       150   118453.879622     5470.655170    1.500331e-02
       200   119482.649137     5908.345043    2.701433e-02
       250   121088.258207     4812.993303    3.119359e-02
       300   121451.395033     4757.045003    3.370962e-02
       350   122610.786651     3849.384289    3.577142e-02
       400   122856.980147     3727.796565    3.679200e-02
       450   122934.912591     3870.214222    3.859678e-02
       500   123454.928696     3463.768837    3.952698e-02

  Elapsed cpu time: 6.630000 seconds.

MD
```

Normal end of execution.

12 September 2017 09:30:21 PM

real 0m6.648s
user 0m6.640s
sys 0m0.002s

c557-903$ export OMP_NUM_THREADS=16

c557-903$ time ./rose_md_OpenMP 2 500 500 0.01
12 September 2017 09:30:28 PM

MD
C version
A molecular dynamics program.

ND, the spatial dimension, is 2
NP, the number of particles in the simulation, is 500
STEP_NUM, the number of time steps, is 500
DT, the size of each time step, is 0.010000

At each step, we report the potential and kinetic energies.
The sum of these energies should be a constant.
As an accuracy check, we also print the relative error
in the total energy.

Step	Potential Energy P	Kinetic Energy K	(P+K-E0)/E0 Relative Energy Error
0	122092.740143	0.000000	0.000000e+00
50	118856.083102	3379.435004	1.169422e-03
100	118817.342909	4258.011058	8.048094e-03
150	118453.879622	5470.655170	1.500331e-02
200	119482.649137	5908.345043	2.701433e-02
250	121088.258207	4812.993303	3.119359e-02
300	121451.395033	4757.045003	3.370962e-02
350	122610.786651	3849.384289	3.577142e-02
400	122856.980147	3727.796565	3.679200e-02
450	122934.912591	3870.214222	3.859678e-02
500	123454.928696	3463.768837	3.952698e-02

Elapsed cpu time: 14.940000 seconds.

MD
  Normal end of execution.

12 September 2017 09:30:29 PM

real 0m1.074s
user 0m14.742s
sys 0m0.210s