Exercise-2

Serial code (`./trainingIPT/exercises/exercise2/example2.c`):

```
#include <stdio.h>
#include <sys/time.h>

#define N 30000

int main(){
  int i, j;
  double x[N+2][N+2], y[N+2][N+2], tmp[N+2][N+2];
  double sum=0;

  //for timing the code section
  struct timeval start,end;
  float delta;

  for(i=0; i <= N+1; i++){
     for(j=0; j <= N+1; j++){
         x[i][j] = (double) ((i+j)%3) - 0.9999;
         y[i][j]= x[i][j] + 0.0001;
     }
  }

  //start timer and calculation
  gettimeofday(&start, NULL);

  for(j=1; j<N+1; j++){
    for(i=1; i<N+1; i++ ){
       tmp[i][j] = 0.167 * (x[i][j] + x[i-1][j] + x[i+1][j] + x[i][j-
1] + x[i][j+1] + y[i+1][j]);
       y[i][j] = tmp [i][j];
       sum = sum + tmp[i][j];
    }
  }

  //stop timer and calculation
  gettimeofday(&end, NULL);
  delta = ((end.tv_sec-start.tv_sec)*1000000u + end.tv_usec-
start.tv_usec)/1.e6;
  printf("\nThe total sum is: %lf\n", sum);
  //print time to completion
```

Ritu Arora 9/13/2017 4:08 AM
Comment [1]: Candidate for parallelization but this loop has ante-dependence - WAR hazard - the value of y in "i+1" iteration is needed in the "i" the iteration.

Ritu Arora 9/13/2017 4:13 AM
Comment [2]: This can result in different answers in different runs of the code.

```
  printf("run time    = %fs\n", delta);
  return 0;
}
```

Breaking the Ante-dependency in the hotspot for parallelization - the refactored Code is shown below

```
#include <stdio.h>
#include <sys/time.h>

#define N 30000

int main(){
  int i, j;
  double x[N+2][N+2], y[N+2][N+2], tmp[N+2][N+2];
  double sum=0;

  //for timing the code section
  struct timeval start,end;
  float delta;

  for(i=0; i <= N+1; i++){
     for(j=0; j <= N+1; j++){
         x[i][j] = (double) ((i+j)%3) - 0.9999;
         y[i][j]= x[i][j] + 0.0001;
     }
  }

  //start timer and calculation
  gettimeofday(&start, NULL);

  for(j=1; j<N+1; j++){
    for(i=1; i<N+1; i++ ){
       tmp[i][j] = 0.167 * (x[i][j] + x[i-1][j] + x[i+1][j] + x[i][j-
1] + x[i][j+1] + y[i+1][j]);
    }
  }

  for(j=1; j<N+1; j++){
    for(i=1; i<N+1; i++ ){
       y[i][j] = tmp [i][j];
       sum = sum + tmp[i][j];
    }
  }
  //stop timer and calculation
  gettimeofday(&end, NULL);
  delta = ((end.tv_sec-start.tv_sec)*1000000u + end.tv_usec-
start.tv_usec)/1.e6;
  printf("\nThe total sum is: %lf\n", sum);
  //print time to completion
  printf("run time    = %fs\n", delta);
  return 0;
}
```

Ritu Arora 9/13/2017 4:12 AM
Comment [3]: We have split the for-loop with ante-dependency between statements into two for-loops

Parallelization Using IPT

login3$ idev

c557-601$ source runBeforeIPT.sh

c557-402$ ../../IPT .solutions/example2_refactored.c

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
1

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 (i = 0; i <= 30000 + 1; i++) {
  for (j = 0; j <= 30000 + 1; j++) {
    x[i][j] = (((double )((i + j) % 3)) - 0.9999);
    y[i][j] = (x[i][j] + 0.0001);
  }
}
```

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

OK - will find the next loop if available.

```
for (j = 0; j <= 30000 + 1; j++) {
  x[i][j] = (((double )((i + j) % 3)) - 0.9999);
  y[i][j] = (x[i][j] + 0.0001);
}
```

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

OK - will find the next loop if available.

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 (j = 1; j < 30000 + 1; j++) {
  for (i = 1; i < 30000 + 1; i++) {
    tmp[i][j] = (0.167 * (((((x[i][j] + x[i - 1][j]) + x[i + 1][j]) + x[i][j - 1]) + x[i][j + 1]) + y[i + 1][j]));
  }
}
```
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)
n

IPT is unable to perform the dependency analysis of the array named [tmp] 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

Ritu Arora 9/13/2017 4:20 AM
Comment [4]: This question helps IPT decide if the array should be part of the shared clause, or firstprivate/private clause.

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)
y

Please choose the function that you want to parallelize from the list below
1 : main
1

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 (i = 0; i <= 30000 + 1; i++) {
  for (j = 0; j <= 30000 + 1; j++) {
    x[i][j] = (((double )((i + j) % 3)) - 0.9999);
    y[i][j] = (x[i][j] + 0.0001);
  }
}
```
Is this the for loop you are looking for?(y/n)
n

OK - will find the next loop if available.

```
for (j = 0; j <= 30000 + 1; j++) {
  x[i][j] = (((double )((i + j) % 3)) - 0.9999);
  y[i][j] = (x[i][j] + 0.0001);
}
```

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

OK - will find the next loop if available.

```
for (i = 1; i < 30000 + 1; i++) {
  tmp[i][j] = (0.167 * (((((x[i][j] + x[i - 1][j]) + x[i + 1][j]) + x[i][j - 1]) + x[i][j + 1]) + y[i + 1][j]));
}
```

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

OK - will find the next loop if available.

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 (j = 1; j < 30000 + 1; j++) {
  for (i = 1; i < 30000 + 1; i++) {
    y[i][j] = tmp[i][j];
    sum = (sum + tmp[i][j]);
  }
}
```

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. j type is int
2. sum type is double

```
2

Please enter the type of reduction you wish for variable [sum]
1. Addition
2. Subtraction
3. Min
4. Max
5. Multiplication
1

IPT is unable to perform the dependency analysis of the array named [ y ] 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

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

```
c557-204$ ls -ltr
total 740
-rw------- 1 rauta G-25072    932 Sep 12 14:04 example2.c
-rwx------ 1 rauta G-25072  20379 Sep 12 14:04 example2
-rw------- 1 rauta G-25072   2090 Sep 12 14:04 ex2_serial.f90
-rw-r--r-- 1 rauta G-25072   1357 Sep 12 16:16 rose_example2_refactored_OpenMP.c

c557-204$ ml intel

c557-204$ icpc -qopenmp -o rose_example2_refactored_OpenMP
rose_example2_refactored_OpenMP.c

c557-204$ ls -ltr
total 741
```

```
-rw------- 1 rauta G-25072    932 Sep 12 14:04 example2.c
-rwx------ 1 rauta G-25072  20379 Sep 12 14:04 example2
-rw------- 1 rauta G-25072   2090 Sep 12 14:04 ex2_serial.f90
-rw-r--r-- 1 rauta G-25072   1357 Sep 12 16:16 rose_example2_refactored_OpenMP.c
-rwxr-xr-x 1 rauta G-25072  26840 Sep 12 16:52 rose_example2_refactored_OpenMP


c557-204$ export OMP_NUM_THREADS=1

c557-204$ time rose_example2_refactored_OpenMP

The total sum is: 105210.000000
run time    = 39.496216s

real    0m47.724s
user    0m38.508s
sys     0m9.110s

c557-601$ export OMP_NUM_THREADS=16

c557-601$ time rose_example2_refactored_OpenMP

The total sum is: 105210.000000
run time    = 11.123440s

real    0m19.439s
user    2m7.395s
sys     0m13.298s
```