

Exercise-1

Serial code (./trainingIPT/exercises/exercise1/example1.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], sum, tmp;

    //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) - 0.99999;
        }
    }

    printf("\nMemory allocation done successfully\n");
    //start timer and calculation
    gettimeofday(&start, NULL);
    for(j=1; j<N+1; j++){
        for(i=1; i<N+1; i++){
            tmp = 0.2 * (x[i][j] + x[i-1][j] + x[i+1][j] + x[i][j-1] +
            x[i][j+1]);
            y[i][j] = tmp;
            sum = sum + tmp;
        }
    }

    //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 2:58 AM

Comment [1]: hotspot for parallelization

Parallelization Using IPT

login3\$ `idev`

c557-601\$ `source runBeforeIPT.sh`

c557-601\$ `../../IPT example1.c`

`"/scratch/01698/rauta/trainingIPT/exercises/exercise1/example1.c", line 8: warning:`

`variable "y" was set but never used`

`double x[N+2][N+2], y[N+2][N+2], sum, tmp;`

^

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

Ritu Arora 9/13/2017 2:28 AM

Comment [2]: Get access to a compute node

Ritu Arora 9/13/2017 2:28 AM

Comment [3]: Setup the environment

Ritu Arora 9/13/2017 2:29 AM

Comment [4]: Path to the "IPT" executable

Ritu Arora 9/13/2017 2:29 AM

Comment [5]: Input source code

Ritu Arora 9/13/2017 2:23 AM

Comment [6]: These warning messages are related to compiling from your code.

Ritu Arora 9/13/2017 2:33 AM

Comment [7]: OpenMP trivia

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;  
  }  
}
```

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;  
}
```

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 = (0.2 * (((x[i][j] + x[i - 1][j]) + x[i + 1][j]) + x[i][j - 1]) + x[i][j + 1]));  
    y[i][j] = tmp;  
    sum = (sum + tmp);  
  }  
}
```

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. tmp type is double
2. j type is int
3. sum type is double

3

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

```
c557-601$ ls -ltr
```

```
total 12
```

```
-rw----- 1 rauta G-25072 891 Sep 12 14:04 example1.c
```

```
-rw----- 1 rauta G-25072 1775 Sep 12 14:04 ex1_serial.f90
```

```
-rw-r--r-- 1 rauta G-25072 1143 Sep 12 14:18 rose_example1_OpenMP.c
```

Ritu Arora 9/13/2017 2:52 AM

Comment [8]: IPT runs some consistency checks

c557-601\$ `ml intel`

c557-601\$ `icpc -qopenmp -o rose_example1_OpenMP rose_example1_OpenMP.c`

c557-601\$ `ls -ltr`

```
total 40
-rw----- 1 rauta G-25072  891 Sep 12 14:04 example1.c
-rw----- 1 rauta G-25072 1775 Sep 12 14:04 ex1_serial.f90
-rw-r--r-- 1 rauta G-25072 1143 Sep 12 14:18 rose_example1_OpenMP.c
-rwxr-xr-x 1 rauta G-25072 26306 Sep 12 14:20 rose_example1_OpenMP
```

c557-601\$ `export OMP_NUM_THREADS=1`

c557-601\$ `time ./rose_example1_OpenMP`

Memory allocation done successfully

The total sum is: 90000.000000

run time = 38.340500s

real 0m43.125s

user 0m37.199s

sys 0m5.883s

c557-601\$ `export OMP_NUM_THREADS=16`

c557-601\$ `time ./rose_example1_OpenMP`

Memory allocation done successfully

The total sum is: 90000.000000

run time = 11.794791s

real 0m16.701s

user 2m19.664s

sys 0m7.835s

Ritu Arora 9/13/2017 2:53 AM

Comment [9]: Load the Intel compiler

Ritu Arora 9/13/2017 3:13 AM

Comment [10]: Compiling the generated code

Ritu Arora 9/13/2017 3:13 AM

Comment [11]: Setting the number of OpenMP threads to use

Ritu Arora 9/13/2017 2:30 AM

Comment [12]: Running the code with the "time" command to see the overall run-time

Ritu Arora 9/13/2017 3:14 AM

Comment [13]: Running the executable

Ritu Arora 9/13/2017 3:14 AM

Comment [14]: Setting the number of OpenMP threads to use

Ritu Arora 9/13/2017 3:14 AM

Comment [15]: Running the executable