MYE023–Parallel Systems and Programming

Programming Assignment #1

My Name
S/N: 12345
E-mail: cs012345@uoi.gr

The first programming assignment is about parallel programming in the shared address space model using OpenMP. We have to parallelize two applications etc...

All experiments where performed and measured in the following departmental system:

PC name	opti7020ws111
CPU	Intel i5-4590
# cores	4
Compiler	gcc v4.8.4

Problem #1

The problem

In this exercise we have to parallelize a matrix-matrix multiplication code etc...This must be done by parallelizing each of the three loops etc...Finally, we are asked to try different schedules.

Parallelization method

We used the sequential program given in the course web page. In order to parallelize the first loop, I simply added the following directive

```
#pragma omp parallel for private(j,k,sum)
```

before the first `i`-loop. Variables `j`, `k` and `sum` must be private so that every thread computes using its own memory space, without affecting the other threads. On the other hand, arrays `A`, `B` and `C` must be shared. In this application, there is no need for mutual exclusion because different threads affect different elements of `C`.

Similarly, for the second `j`-loop we ...

The parallelization of the third `k`-loop, however, ...

Experiments and measurements

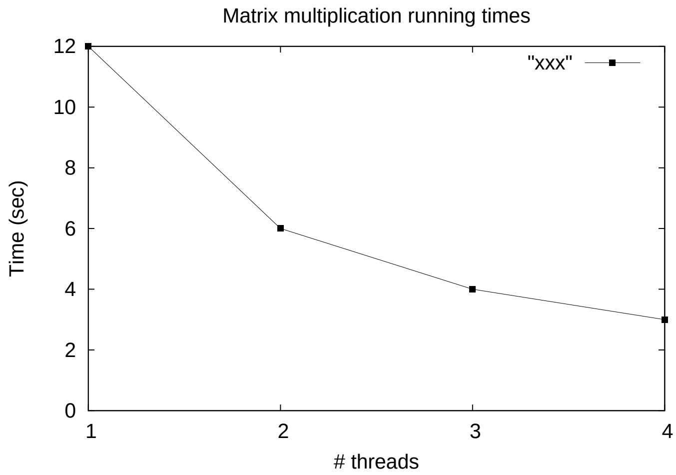
All programs were executed on the system we mentioned in the introduction. Timing was performed using the `omp_get_wtime()` call ...

We used from 1 to 4 threads and ...

Every experiment was performed 4 times and we calculated the average execution time. Care was taken not to include the time required for reading the files from disk. The following table has the results (all times in *sec*):

Threads	Execution 1	Execution 2	Execution 3	Execution 4	Average
1	11.9	12.0	12.0	12.1	**12.0**
2	5.9	6.0	6.0	6.1	**6.0**
3	3.9	4.0	4.0	4.1	**4.0**
4	2.9	3.0	3.0	3.1	**3.0**

Based on the above measurements we obtain the following plot:



BE CAREFUL IN THE PLOTS:

1. *We are interested only in* ***average times****, not the times of a specific execution!!!*
2. *The X-axis and the Y-axis must have a title that* ***specifies the quantity it represents****, for example '# threads' or 'Time'.*
3. *The* ***units of measurement*** *(e.g. sec) must be shown, either on the axes titles or in the plot legend.*

Comments

(Here you have to write a) your observations, i.e. what you saw and b) your explanations, i.e. why you had those measurements)

Based on the results, we see that as the number of threads increases, we have a significant reduction in execution times which is almost ideal (i.e. k threads run approximately in a fraction $1/k$ of the sequential execution time). Also, the choice of schedule affected ... **[observations]**

All the above were actually expected since all iterations were split evenly among the threads and all iterations have the same computational load. Consequently ... **[explanations]**

Problem #2

The problem

...

Parallelization method

...

Experiments and measurements

...

Comments

...