Exercise 1: Approximate Pi

Gabriele Sarti

May 13, 2019

Abstract

In the first exercise for the course of Parallel computing, our aim is to approximate the value of π using the Montecarlo integral approximation.

1 Procedure and Results

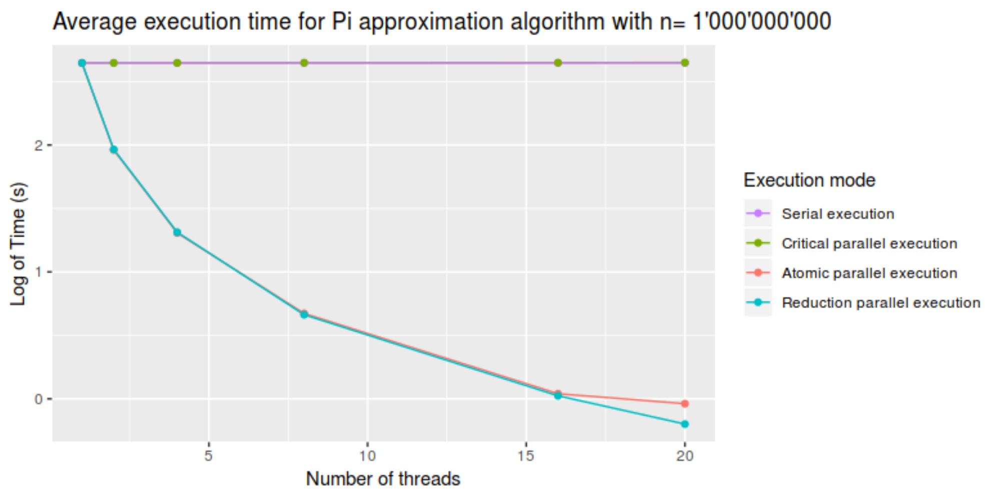
I implemented the approximation algorithm in four different modes, as required by the assignment: a serial version, a parallel version using the "critical" keyword, a parallel version using the "atomic" keyword and a parallel version using the "reduction" keyword. All parallel versions are using OpenMP for multithreading.

I measured the execution time of all versions on a Ulysses node using 1, 2, 4, 8, 16 and 20 threads to see the scalability of my implementations, with a problem size of 1'000'000'000 using the gcc compiler with optimization -O0. Table 1 and Figure 1 show the results obtained through this procedure.

Table 1: Results of Exercise 1 execution

# threads	Serial time (s)	Critical time (s)	Atomic time (s)	Reduction time (s)
1	14.10	14.10	14.10	14.10
2	14.11	14.11	7.10	7.13
4	14.10	14.11	3.70	3.72
8	14.10	14.13	1.96	1.94
16	14.11	14.14	1.04	1.02
20	14.11	14.15	0.96	0.81

Figure 1: Scalability of Montecarlo integral approximation for different number of threads.



From the plot and the data it is evident that parallelizing the code leads to significant improvements in terms of execution time. Parallelization using critical keyword doesn't show any improvement since the function that should be made parallel is executed in a serial fashion to comply to the keyword.

It is also interesting to note that the performances of atomic and reduction keyword are quite similar in terms of performance, with a slight improvement visible for higher thread counts.

2 Reproducibility

In order to obtain results that are similar to those listed above, simply clone the Github repository [1] in the personal Ulysses folder and run the following command from your main folder:

```
qsub -q reserved3 -l nodes=1:ppn=20 parallel-computing/Assignments/ex01/ex1.sh
```

Results will be different since the Makefile has been modified to speed up the computation by using mpicc compile with optimization level -O3. See exercise 3 for updated OpenMP values. The result will be contained in the ex1.sh.o* file which will be created in the Assignments folder.

References

[1] https://github.com/gsarti/parallel-computing