Algorithm Engineering Lab Assignment 3

Brian Zahoransky (brian.zahoransky@uni-jena.de)

January 15, 2021

1. How does the ordered clause in OpenMP work in conjunction with a parallel for loop?

Using the ordered clause, it seems somehow like the for loop is executed serially. In the lecture was the following example shown. A for loop iterates over the variable i. The variable is initialized with 0 and is increased by one each iteration. The loop stops when i has reached 10,000. The goal is to find a number between 133 and 140. This is achieved by checking if i fulfills the condition. When this happens, i is written into the result variable and the algorithm skips the next iterations. If the for loop is executed parallel without the ordered clause, the result will be some number between 133 and 140. On the other hand, if the ordered clause is placed right before i is written into result, the result is the smallest number, 134. In this case the whole for loop is executed in parallel except the lines when i is written into the result. These lines are executed as in a serial version.

2. What is the collapse clause in OpenMP good for?

This clause allows combining nested for loops. It can be useful in the following example. Consider a three times nested for loop each having four iterations. Furthermore, it is assumed that the executing machine has 16 physical cores. To use all the cores, the first, second, and optionally also the third for loop could be parallelized by insert #**pragma omp parallel for** in front of each for loop. Alternatively, **collapse(3)** could be added to the first omp pragma while removing the other two pragmas to achieve the same result. The collapse clause allows to combine any number of for loops by replacing the 3 in **collapse(3)**.

3. Explain how reductions work internally in OpenMP.

First, each thread creates a local copy of the variable which will be reduced. Note that the initialization depends on the operation. For a summation the variable is initialized with zero, whereas for a product it is initialized with one. While the threads are running the local copies are updated. To perform the reduction, the local copies are combined with the original value.

4. What is the purpose of a barrier in parallel computing?

The purpose of barriers is synchronization. All threads stop at a barrier as long as the last thread has also reached the barrier. This may be useful if all threads need the results of the other threads.

5. Explain the differences between the library routines:

omp_get_num_threads(): returns the number of threads in a parallel region
omp_get_num_procs(): returns the number of logical cores in the machine
omp_get_max_threads(): returns the maximal number of threads, which could be started currently

6. Clarify how the storage attributes private and firstprivate differ from each other.

Firstprivate creates for each thread an initialized copy of the given variable.
Private creates for each thread an uninitialized copy of the given variable.

7. Write in pseudo code how the computation of π can be parallelized with simple threads.

Have a look at 7th_exercise.py program.