Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences
Fachbereich Informatik
Lehrgebiet Programmierung komplexer Systeme
Prof. Dr. Rudolf Berrendorf

Parallel Systems
WS 19/20
given out : 02.12.2019
to deliver : 15.12.2019

Assignment 5 : Scheduling

Assignment 1 (20 points) :
On the web page you find for this assignment a test framework for loop scheduling algorithms. In `sched.c` you find several loop scheduling algorithms implemented or partially implemented, in `sched_test.c` the test framework that tests all available scheduling algorithms with different workloads and print out timings.

Implement the following loop scheduling algorithms in `sched.c`:

- Self Scheduling
- GSS
- Factoring (first version)

The corresponding function prototypes exist already in that file. For these dynamic algorithms you must use adequate synchronization primitives for efficient synchronization (e.g. mutual exclusion) between the participating threads. For simplicity you can use for example OpenMP synchronization constructs like `critical, atomic, ...`

Running the program `./sched_test.exe`, executes an OpenMP parallel test program with parallel loops that are scheduled according to the scheduling algorithms (i.e., the original OpenMP scheduling constructs are not used). The program tests the scheduling functions

- with four different work loads (constant/ increasing / decreasing / random) with increasing iteration numbers (see corresponding slide of lecture)
- with three different variations of number of iterations / work inside iterations (few iterations with heavy workload up to many iterations with light workload in each iterations)

simulating different parallel loops / work loads. For correctness tests of your modifications you can use `wr0` interactively. For timings, run the program in a batch job on `wr43` (see corresponding job script) with all available processors on that computer (i.e. 96) and compare the results you got for the different algorithms. The test framework checks also after every run of a parallel loop for correct execution of that parallel loop. With the initial program version without your modifications for the three algorithms you will see therefore error messages for the dynamic scheduling tests.

The test framework is an OpenMP program that starts a parallel region and executes the scheduling algorithms in parallel. The code that simulates the parallel loop and that is executed by all OpenMP threads in parallel looks like this:

```
#pragma omp parallel
{
    // ...
    setup_alg(n, p, iam);
    while (sched_alg (&start_iteration , &end_iteration , n, p, iam))
        {
          /* do work on chunk assigned by scheduling algorithm */
          for (int i = start_iteration; i <= end_iteration; i++)
              /* do the work */
              work_fun(i);
        }
    // ...
}
```

Before a loop gets scheduled, each time a setup function `setup_alg` for that algorithm is called where a scheduling function may do any internal setups, e.g., compute any constants. `n` is the number of iterations to be scheduled, `p` the number of participating processors, and `iam` the own processor / thread number.

Then every scheduling algorithm `sched_alg` gets one or more times called and it has to deliver in `start_iteration` and `end_iteration` the iteration intervall that the thread that made the call should execute. If there are no more iterations left for the calling thread the scheduling function must return `false`, else `true`.

As an example how to realize a scheduling algorithm, you may have a look at the static scheduling algorithms implemented in the file `sched.c`. But be aware that these static scheduling algorithms don't need any synchronization. See also the additional hints at the beginning of that file.