Parallel Programming WS15 HOMEWORK #9

Submit by email to: `fabregat@aices.rwth-aachen.de`
Due by: Friday, January 22th, 5pm.

OpenMP – Part 2

1. **Task 1: Scheduling**

   In this task you will reason about and test different scheduling schemes in OpenMP. We provide you with the program `HW9.1-scheduling.c`, which runs a number of *for* loops, each with a different distribution of workload per iteration. More specifically, the program contains 4 independent (non-nested) *for* loops, each of them with the following type of distribution of work:

   - **Constant**: all iterations perform the same amount of work.
   - **Small fluctuations**: all iterations perform a similar amount of work, with fluctuations of a small percent.
   - **Large fluctuations**: most iterations perform little work, while a few of them, concentrated in the beginning of the loop, may require several orders of magnitude more work than the baseline.
   - **Increasing**: each iteration performs an amount of work proportional to the index of the loop, which ranges from 0 to some n.

   Each work distribution is simulated by four different functions that execute the necessary amount of work in each case, based on the iteration number i and the number of iterations n.

   Before proceeding to experimentally observe the behavior for different scheduling schemes, we ask you to look into the code of each of these functions, and produce four plots illustrating the work distribution. Represent the iteration number in the x-axis and the amount of work in the y-axis. As an example, the plot for the *constant* distribution is a horizontal line. No need for wasting time in fancy plots, you can use any plotting program or even handmade and scanned plots :)

   Then, still before running any code, you must argue which between `static` and `dynamic` schemes with different *chunk sizes* should be faster in each case. Explain your choice in a few lines.

   Finally, run the code and compare with your predictions. In case you were mistaken in any of your predictions, try to explain why this was the case. To run the code, we provide you with a *script* which you can run by simply typing

   ```
   ./HW9.1-script
   ```

   in you shell. Following the example in the script, run the code with different schemes and chunk sizes.

   We suggest you run the job in the university's cluster using the *jobscript* provided:

```
bsub < HW9.1-jobscript
```

By default, the script uses 4 threads. Feel free to try with a larger number of threads. If you use more than 8 threads, remember to change the line "`#BSUB -n 8`" in the jobscript to request a larger node.

2. **Task 2: Data-sharing attributes and synchronization**

   In this task you will further practice with data-sharing attributes, race conditions, and synchronization constructs. We provide you with the program `HW9.2-mandel.c`[1] which contains at least 4 bugs. We ask you to identify and fix these bugs. We also ask you to list the bugs you found and explain which problem they were causing and how you fixed them. Instructions on how to run the program and the expected result are given at the beginning of the file. Test with different number of threads.

[1]This program was written by Mark Bull and Tim Mattson and is often used in OpenMP tutorials to illustrate a number of common bugs.