HPC Laboratory Assignment
Lab 4
OpenMP (Part I)

J.R Herrero and M.A. Senar

May 17, 2024

Contents

4 OpenMP tutorial examples **2**
4.1 OpenMP basics and loop parallelism 2
4.2 Deliverable 2
4.3 OpenMP questionnaire 3

Note:

- All files necessary to do this laboratory assignment are available in `Escritorio/alumnos/OpenMP`. From the the root of your home directory unpack the files with the following command line: `"tar -zxvf openmp_examples.tar.gz"`.

Session 4

OpenMP tutorial examples

This lab session has been prepared with the purpose of guiding you through a set of very simple examples that will be helpful to practice the main components of the OpenMP programming model, filling-in the questionnaire in of the deliverable laboratory assignment. In order to follow them, you will need:

- The set of files inside the `Escritorio/alumnos/OpenMP` directory.
- The set of slides about OpenMP available through the course web.

4.1 OpenMP basics and loop parallelism

1. Open each of the codes in the directory where files have been decompressed and answer the questions in the questionnaire associated to it; you can compile (e.g. `"make parallel"`) and run (e.g. `"./parallel"` to check your answers.
2. Consult "`OpenMP Basics`" and "`Loop Parallelism in OpenMP`" of the tutorial slides, if necessary.

4.2 Deliverable

After the session for this laboratory assignment, and before starting the next one, you will have to deliver a **report** in `PDF` format (other formats will not be accepted) containing the answers to the following questions. Your professor will open the assignment at the course website and set the appropriate dates for the delivery. Only one file has to be submitted per group through the course website.

Important:

- Please, follow the same recommendations that we made for previous deliverables.
- In the front cover of the document, please clearly state the name of all components of the group, the identifier of the group (username `biohpc-XX`), title of the assignment, date, academic course/semester, ... and any other information you consider necessary. As part of the document, you can include any code fragment you need to support your explanations.

4.3 OpenMP questionnaire

When answering to the questions in this questionnaire, please DO NOT simply answer with yes, no or a number; try to minimally justify all your answers. Answer the questions in this document, not the ones in the source files that may be outdated. Sometimes you may need to execute several times in order to see the effect of data races in the parallel execution.

hello.c

1. How many times will you see the `"Hello world!"` message if the program is executed with `"./hello"`?
2. Without changing the program, how to make it to print 4 times the `"Hello World!"` message?

how_many.c: Assuming the `OMP_NUM_THREADS` variable is set to 8 with `"export OMP_NUM_THREADS=8"`

1. How many `"Hello world ..."` lines are printed on the screen? Why?
2. Which mechanism overrides the number of threads already set by other mechanisms?

hello_world.c: Assuming the `OMP_NUM_THREADS` variable is set to 8 with `"export OMP_NUM_THREADS=8"`

1. Is the execution of the program correct? (i.e., prints a sequence of `"(Thid) Hello (Thid) world!"` being `Thid` the thread identifier) Which data sharing clause should be added to make it correct?.
2. Are the lines always printed in the same order? Could the messages appear intermixed?

data_sharing.c

1. Which is the value of variable `x` after the execution of each parallel region with different data-sharing attribute (`shared`, `private` and `firstprivate`)?
2. What needs to be changed/added/removed in the first directive to ensure that the value after the first parallel is always 8?.

parallel.c

1. How many messages the program prints? Which iterations is each thread executing?
2. Change the for loop to ensure that its iterations are distributed among all participating threads.

datarace.c (execute several times before answering the questions)

1. Is the program always executing correctly?
2. Add two alternative directives to make it correct. Explain why they make the execution correct.

barrier.c

1. Can you predict the sequence of messages in this program? Do threads exit from the barrier in any specific order?

for.c

1. How many and which iterations from the loop are executed by each thread? Which kind of `schedule` is applied by default?
2. Which directive should be added so that the first `printf` is executed only once by the first thread that finds it?.

schedule.c

1. Which iterations of the loops are executed by each thread for each `schedule` kind?

nowait.c

1. How does the sequence of `printf` change if the `nowait` clause is removed from the first `for` directive?
2. If the `nowait` clause is removed in the second `for` directive, will you observe any difference?

nested_for.c

1. How many loop iterations are executed in the program by each thread? Variables `i` and `j` are private. Why?
2. Modify the program to distribute only the iterations of the outermost loop. Which loop iterations are executed by each thread now?
3. Modify the program to distribute only the innermost iterations. Which loop iterations are executed by each thread now?

collapse.c

1. Which iterations of the loop are executed by each thread when the `collapse` clause is used?
2. Is the execution correct if the `collapse` clause is removed? Which clause (different than `collapse`) should be added to make it correct?.

ordered.c

1. Can you explain the order in which `printf` appear?
2. How can you ensure that a thread always executes two consecutive iterations in order during the execution of the ordered part of the loop body?

doacross.c

1. In which order are the "Outside" and "Inside" messages printed?
2. In which order are the iterations in the second loop nest executed?
3. What would happen if you remove the invocation of `sleep(1)`. Is the order of the iterations modified with respect to the previous case? Execute several times to answer in the general case.