Lab 2: Brief tutorial on OpenMP programming model

par4111
Adrià Cabeza, Xavier Lacasa
Departament d' Arquitectura de Computadors

March 16, 2019
2018 - 19 PRIMAVERA

Contents

1 Introduction

In this session we will learn about the OpenMP programming model.

2 Parallel regions

2.1 hello.c

1. How many times will you see the "Hello world!" message if the program is executed with *./1.hello*?

We see the message 24 times. This is due to the *#pragma omp parallel* call before *printf("Hello world! \n");*, which makes every available thread execute the *printf*. In Boada 1, they happen to be 24 threads, and so the message is printed 24 times.

2. Without changing the program, how to make it to print 4 times the *Hello World!* message?

By setting the number of threads available to 4, only 4 threads would execute the *printf("Hello world! \n");* line and so the message would be displayed only 4 times.

2.2 2.hello.c

1. Is the execution of the program correct? (i.e., prints a sequence of *(Thid) Hello (Thid) world!* being Thid the thread identifier). If not, add a data sharing clause to make it correct?

2. Are the lines always printed in the same order? Why the messages sometimes appear intermixed? (Execute several times in order to see this).

2.3 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?

2. What does omp get num threads return when invoked outside and inside a parallel region?

2.4 data_sharing.c

1. Which is the value of variable x after the execution of each parallel region with different datasharing attribute (shared, private, firstprivate and reduction)? Is that the value you would expect? (Execute several times if necessary)

3 Loop parallelism

3.1 schedule.c

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

3.2 2. nowait.c

1. Which could be a possible sequence of printf when executing the program? 2. How does the sequence of printf change if the nowait clause is removed from the first for directive? 3. What would happen if dynamic is changed to static in the schedule in both loops? (keeping the nowait clause)

3.3 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?

4 Synchronization

4.1 datarace.c

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

4.2 barrier.c

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

4.3 ordered.c

1.Can you explain the order in which the *Outside* and *Inside* messages are printed? 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?

5 Tasks

5.1 1.single.c

1. Can you explain why all threads contribute to the execution of instances of the single worksharing construct? Why are those instances appear to be executed in bursts?

5.2 fibtask.c

1. Why all tasks are created and executed by the same thread? In other words, why the program is not executing in parallel? 2. Modify the code so that the program correctly executes in parallel, returning the same answer that the sequential execution would return.

5.3 synchtasks.c

1. Draw the task dependence graph that is specified in this program 2. Rewrite the program using only taskwait as task synchronisation mechanism (no depend clauses allowed)

5.4 taskloop.c

1. Find out how many tasks and how many iterations each task execute when using the grainsize and num tasks clause in a taskloop. You will probably have to execute the program several times in order to have a clear answer to this question. 2. What does occur if the nogroup clause in the first taskloop is uncommented?

6 Conclusion