Take-Home Exam PTH_MSORT

Parallel Programming & Architectures

Consideration

- ✓ Your code is automatically graded using a script, and therefore, if your file/folder names are wrong you will receive a grade of **zero**. Please read and follow the instructions carefully. Common mistakes include
  - Different file or folder names
  - Different formatting of input or output
  - Not paying attention to case sensitiveness of C++ and Linux
- ✓ Go to the folder ~/the/pth_msort/ in your home directory on the server and put your codes in this directory and remove any compiled binaries and test cases.
- ✓ Make sure your code compiles and runs without any error **on the server**. Your grade will be **zero** if any compile or runtime error occurs on the server. **Any!**
- ✓ The provided test cases, examples and sample codes (if any) are only to better describe the question. They are **not** meant for debugging or grading. It is your responsibility to think of and generate larger and more complex test cases (if necessary) in order to make sure your software works correctly for all possible scenarios.
- ✓ Start early and don't leave everything to the last minute. Software debugging needs focus and normally takes time.
- ✓ Just leave your final programs on the server. **Don't** email anything!
- ✓ Your grade is divided into several parts. In all cases, if you miss **correctness** (i.e. your code doesn't satisfy desired functionality), you miss other parts (e.g. speed, coding style, etc.) too. This rule is applied separately for each section of a take-home exam. So for example, in cuda_mm, your code might not be correct for M >= x but still you will get your grade for lower M values.
- ✓ Talking to your friends and classmates about this take-home exam and sharing ideas are *OK*. Searching the Internet, books and other sources for any code is also *OK*. However, copying another code is **not OK** and will be automatically detected using a similarity check software. In such a case, grades of the copied parts are **multiplied by -0.5**. Your work must be 100% done only by yourself, and you **should not** share parts of your code with others or use parts of other's codes. Online resources and solutions from previous years are part of the database which is used by the similarity check software.

pth_msort.c – Parallel merge sort algorithm in pthread

Grade: 20% correctness, 80% speed

In this take-home exam, we are going to parallelize merge sort algorithm. For this purpose, follow these steps:

A) Use 4 thread of executions to perform 4 'merge' jobs in parallel. Use pthread library for this purpose.
Being more explicit, regarding Fig.1, you should launch one thread for each of the 4 sort modules, i.e., total of 4 threads for all sort modules. Each sort module is itself a merge sort. Your program must work for any array size N=2^M (24 <= M <= 30).
2 middle merge modules: Use serial merge algorithm for the 2 middle merge modules. You should launch one thread for each of the 2 middle merge modules, i.e., total of 2 threads for all middle merge modules.

B) Next let's going deeper parallelize the last 'merge' itself. You must parallelize the last merge module using 4 threads in this step.

Write your program in C (not C++) using Linux pthread library. Use the provided pth_msort.c to start your work.

Use the provided sample codes to start your work. Only modify pth_msort.c file and do not change other files. Create the threads inside msort_pth.c file, function mergeSortParallel. Put this file in determined path in your home directory on the server. All other files in that path will be ignored.

Compile: *gcc -O2 pth_msort_test.c pth_msort.c -lpthread -lm*
Execute: *./a.out M*

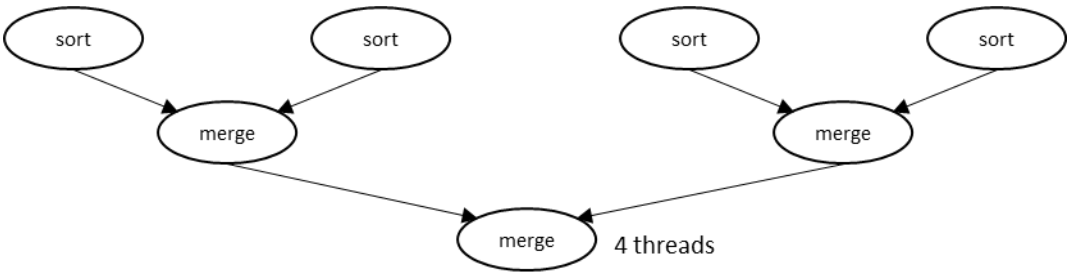

Fig. 1 – Merge sort algorithm