Assignment 3: OpenMP Producer/Consumer

Please read these instructions carefully. Your grade is based upon meeting all requirements stated in this assignment.

Background and Problem Statement

We now have a working pthreads parallel producer/consumer program. Congratulations! We also have them benchmarked for runtime so that we can compare their relative performance with other parallel implementations.

A design objective for OpenMP was to take in-tact serial programs and inject compiler directives which would easily parallelize them. We have seen in class how this is possible in some cases. However, this may or may not be practicalin the general case, nor with your serial consumer/producer program.

Program Requirements

- Start with the skeleton program provided (prod_consumer_orig.c) and then re-architect it as needed to add support for OpenMP based parallelism.
- Your final source code MUST be named prod_consumer_omp.c
- You may read the input from either a serial or a parallel region within your program. All input should be read from stdin as in lab 1.
- You program should support having multiple “producer” threads which will perform populate a shared data structure like a stack or a queue.
  - The simplest way to implement this would be with a stack with one pointer as below.
  - But you are free to choose any other data structure you prefer. The data structure can be either declared globally or locally to the main function.
    - int buffer[MAX_BUF_SIZE] = {0};
    - int location = 0;
- You program should support having multiple “consumer” threads, which will take work from the shared data structure, perform the dummy computations.
  - The consumers MUST consume ALL the data the producers put in the shared data structure. Nothing should be left in the shared data structure once all the consumers have exited. If consumers do not read all the data, you will lose points.
  - We use a usleep to simulate this work.
    - The skeleton file has two values for usleep.
    - One is 10 and another is 100000.
    - Use 10 for development.
    - You must use a value of 100000 when doing the final runs for your submission. If you reduce this because your program took too long to finish/did not finish, it will be considered as a sub-optimal implementation, and you will lose points
- You may perform your output from either a serial or a parallel region within your program.
  - Output should be printed on stdout.
  - The order of output NEED NOT be the same as the order of the input file.
- The skeleton program has some prints for output.

  - The insert_data function or producer should have print of the following form. It is used for data validation by run_script.sh
    - printf("producer %d inserting %d at %d\n", producerno, number, location);
  - The extract_data function or consumer should have print of the following form. It is used for data validation by run_script.sh
    - printf("consumer %d extracting %d from %d\n", consumerno, value, location);
  - The run_script file relies on searching for two strings “inserting” and “extracting” to check for correctness. So please retain these prints or those of similar form.
- The top 10 fastest programs will be subjected to a stress benchmark.
  - The top 5 from that pool will receive extra credits.
- You should try to minimize the number of changes needed.
- In the simplest case, with a data structure of stack and one location variable as given in the skeleton program, you only need 40 lines of change from the skeleton program to implement your OMP-based parallel version of the producer consumer problem. 11 of those lines are OMP pragmas.
  - In the best case, you will only need 3 different types of OMP pragmas.
    - private, parallel, shared, atomic, etc are considered as pragmas
  - There will be extra credits if you can complete your changes with less than 45 lines of change and less than 4 OMP pragmas.

Input Files Provided

The following 4 files have been provided to you. All files are available on Owens in the /fs/ess/PAS2171/ /Lab2-Assignment directory.

- prod_consumer_orig.c – The skeleton file
- Input files – longlist and shortlist
- run_script.sh – A script to run the final program and collate the output

Testing Instructions

- Use the GNU compiler (“gcc”) only.
  - run_script.sh should already do this.
  - Your program should compile without warnings.
  - Points will be deducted for any warnings
- Logging in to Owens using SSH

  ssh -X -C <username>@owens.osc.edu
- When compiling and making short (less than a couple minutes) test runs, you can use the login nodes.
- For final benchmark runs or when running with the larger value of usleep, please do it on an interactive allocation as described below.
- Getting an interactive allocation

  salloc -N <number_nodes> -A PAS2171 --time=<time_in_minutes>

  e.g. To request for one full compute node from the serial partition for 30 minutes, you can use the following command. I am just using 30 as an example. Please use higher values as appropriate.

  -bash-4.2$ salloc -N 1 -A PAS2171 -p serial --time=30

  salloc: Pending job allocation 17868816

salloc: job 17868816 queued and waiting for resources

salloc: job 17868816 has been allocated resources

salloc: Granted job allocation 17868816

salloc: Waiting for resource configuration

salloc: Nodes o0646 are ready for job <- That is the node which was allocated for you

bash-4.2$ hostname

owens-login03.hpc.osc.edu <- You're still on the head node when you get the control back

bash-4.2$ ssh o0646 <- This will move you to the node that was allocated

<….snip….>

-bash-4.2$ hostname Now you're on the node that was allocated.

o0646.ten.osc.edu

Report Requirements

Provide a 1-2 page report which includes the following (note that accuracy and brevity are prized, needlessly lengthy reports may lower your score).

- You will submit the PDF report on Carmen.
- In your PDF report, you should address the following points
  - Provide a brief explanation describing how your changes ensure mutual exclusion.
  - Provide a chart or graph which illustrates the scalability (or lack thereof) of your parallel solution.
    - Y-axis of the graph will represent time taken for the producer consumer to complete as reported by the run_script.sh
    - X-axis of the graph will represent the number of consumers
    - The graph will have one line each for number of producer
  - Among the different combinations of producers and consumers, select ONE as the “best” and explain why you selected the number of producer and consumer threads for your final implementation that you chose.
  - Describe any unusual or unexpected results your found.

Code Submission Instructions

- The run_script.sh file will compile your program and run it for different combinations of producers and consumers with different input files.
- On successful completion, the run_script.sh file will create a sub-directory named “cse5441-omp-lab” and place all necessary files for submission there.
- Submit your cse5441-omp-lab directory from an Owens login node using the OSC submit system
  - More details about how to submit your assignment from OSC is available at the following website: https://www.osc.edu/resources/getting_started/howto/howto_submit_homework_to_repository_at_osc
- Submit this directory to “lab2”
  - You can use the following command to do the submission

- "/fs/ess/PAS2171/CSE-5441-SP22/submit"

– Note that the system will automatically close at the specified deadline for submission. You will not be able to submit your files after that.

– Here is a sample set of steps to submit.

./run_script.sh

Program may be using….

<…snip…>

-bash-4.2$ ll

total 72
-rwxr-xr-x 1 osc0577 PZS0622 14344 Mar 15 09:38 a.out
drwxr-xr-x 2 osc0577 PZS0622 4096 Mar 15 09:42 cse5441-omp-lab
-rw-r--r-- 1 osc0577 PZS0622 9170 Mar 15 09:42 cse5441-omp-lab.tgz
-rw-r--r-- 1 osc0577 PZS0622 410 Mar 15 09:31 longlist
-rw-r--r-- 1 osc0577 PZS0622 9404 Mar 15 09:31 prod_consumer_omp.c
-rw-r--r-- 1 osc0577 PZS0622 8468 Mar 15 09:31 prod_consumer_orig.c
-rwxr-xr-x 1 osc0577 PZS0622 5137 Mar 15 09:38 run_script.sh
-rw-r--r-- 1 osc0577 PZS0622 78 Mar 15 09:31 shortlist

-bash-4.2$ /fs/ess/PAS2171/CSE-5441-SP22/submit

****

Hello. This script will submit your assignment assigned by Hari Subramoni to OSC

Note:

Before you run this script, please create one directory which includes all the files you want to submit

The previous submission of the same assignment from you will be replaced by the new submission

Enter the name of assignment and press [ENTER]

>>>lab2

Enter the absolute path of the directory which includes all the files of your assignment lab2 and press [ENTER]

You will submit the current directory if you simply press [ENTER]

>>>/users/PZS0622/osc0577/CSE-5441/OpenMP/Homework/cse5441-omp-lab

Your assignment lab2 is submitted succesfully. Thank you!

Academic Integrity

Collaborating or completing the assignment with help from others or as a group is NOT permitted

Copying or reusing previous work done by others is not permitted. You may reuse the work you did as part of this class