University of Delaware
Department of Electrical and Computer Engineering
ELEG652 - Principles of Parallel Architectures
Fall 2014
Prof. Guang R. Gao

Assignment 3
Parallel Programming for Shared Memory: POSIX Threads and OpenMP

Due on Saturday Oct. 11th, 23:59

Thread Ring

1. [20%] Create a ring program in which each thread will output its rank to stdout in order. For example: if run with 4 threads, the numbers 0, 1, 2, and 3 should be outputted to stdout, in order, by ranks 0, 1, 2, and 3 respectively. Implement the ring using:

a) [10%] PThreads.
b) [10%] OpenMP.

Matrix-Vector Multiplication

2. [25%] A Matrix-Vector multiplication program is defined as $c = A \times b$: A is a matrix $A_{N,N}$, b is a vector $b_{N,1}$, and C is a vector $c_{N,1}$. Implement a multithreaded Matrix-Vector multiplication, distributing the computation to different threads statically and as evenly as possible. Report the results on stdout. Implement the ring using:

a) [15%] PThreads.
b) [10%] OpenMP.

Dot Product

3. [30%] Implement a multithreaded version of the dot product of two vectors *V1 and V2*: $V1_N \cdot V2_N = \sum_{i=1}^{N} V1_i \times V2_i$. Provide two implementations:

a) [10%] PThreads — Provide two different implementations for PTHREADS: one which uses a mutex, and after reading the documentation of your compiler, one which uses atomic operations. For example, if you use the GNU Compiler Collection (gcc), you can use type __sync_fetch_and_add(type* addr_to_modify, type value, ...) to atomically add a value to a shared variable.
b) [20%] OpenMP — Provide three implementations for this one: first using a critical section, second using an atomic section, and finally using a reduction. Go to http://openmp.org and look for the specification of either OpenMP 3.1 or 4.0 to learn how to express a reduction in a loop.

Linked List Traversal

4. [25%] A linked list is a data structure which chains its elements through a field, usually using pointers or references. It is possible to apply a set of modifications to the elements of a linked list simply by traversing each element of the list, applying the required process, then moving on to the next element in the list. We provide a naïve linked list implementation. Two files (process_list_pthd.c and process_list_omp.c) are provided but need to be completed. An example of list traversal can be found in linked_list.c (function find), as well as in process_list_seq.c. Moreover, process_list_pthd.c features two helper functions, spawn and join, to simplify the use of PTHREADS. After reading the various files of the project, your job is to complete it in two ways:

a) [5%] A certain number of utility functions used in linked_list.c are used, which are defined in utils.h. However we do not provide the file: you must implement the relevant functions, which are meant to simplify error handling.

b) [20%] Complete process_list_pthd.c and process_list_omp.c so that the list processing is parallelized. Use the omp task pragma to provide an OpenMP implementation of process_list. Follow the template provided in the tutorial slides (slides 38-46).

Submission:

Submit a report with your answers using an IEEE Paper Template for the report (http://www.ieee.org/conferences_events/conferences/publishing/templates.html). Remember to cite all your sources.

Include any source files you wrote. Each program you write must be commented and have its own Makefile.

Remember to include the HW/SW specifications of the machine(s) where you run your experiments.

Send all the files as a single ZIP named *<YOUR_NAME>-lab<NUMBER_OF_LAB>-eleg652-14f.zip* (e.g. *johndoe-lab1-eleg652-14f.zip*) to Jaime Arteaga jaime@udel.edu with subject *ELEG652-14F LAB3* before the specified deadline.

If you miss the deadline, you can submit the homework until 17.45 of the following Tuesday, with the homework's total grade being decreased by 10% per day (i.e. homework will be graded over 100% until 23.59 of Friday, 90% until 23.59 of Saturday, 80% until 23.59 of Sunday, 70% until 23.59 of Monday, 60% until 17.45 of Tuesday).