Programming Assignment 5 - hierarchical parallelism

Due Date: Wed., 12/5, 11:59pm

Background

Distributed computing both opens up the possibility of an entire network of computers collaborating simultaneously on an single application while also enabling multiple levels of concurrent parallelism.

Using MVAPICH2, an open source standard MPI implementation, combined with global shared memory threading techniques we will implement a hierarchical parallel version of producer-consumer.

Assignment

Create an MPI producer - consumer program using the global communicator. Your rank 0 MPI process will be your master MPI process. This process will:

- perform all input and output functions
- populate and manage a buffer of work to be performed
- using either pthreads or OpenMP, create four "managing threads" to manage remote MPI processes.
- print work results from a buffer of completed tasks (print requested transform, initial input value, intermediate value α – see below – and final result value.)

Each of your managing threads will:

- obtain the next work unit from the work buffer
  (managing threads should not buffer work – manage one work unit at a time)
- send the work unit to the thread's corresponding remote MPI process
- receive the results from the thread's corresponding remote MPI process
- insert each result into the completed tasks buffer

Each of your remote MPI processes will:

- receive work units, one at a time, from a corresponding managing thread
- perform the following computations:
  - compute the requested transform on the input value, producing intermediate value α
  - add 1, modulo 1000, to intermediate value α, producing intermediate value β
  - perform the requested transform on intermediate value β, producing the final result value for that work unit
- send the requested transform, initial input value, intermediate value α and final result value back to the corresponding managing thread

Write a short report (no more than 3 pages) which provides:

- a summary of the run-time results for your MPI distributed application. For each data file
  - accumulate all the producer time and consumer time spent at all nodes and report both numbers
  - measure the total wall-clock time of your application.
- a summary of your observations and a run-time comparison for all of your producer-consumer implementations.

Input Data Format

Use the input data files PC_data_t00100, PC_data_t01000 and PC_data_t05000 from previous labs. Omitt the largest PC_data_t10000 file from your testing.

Testing and Submission

Please note that OSC run queues, including the interactive queues, may get exceptionally long at this time of the semester

Follow the testing and submission guidelines for lab4, using directory “lab5” for this assignment.

Run your program using 5 nodes (request all cores for all nodes so that you have dedicated processors), one for the master MPI process and managing threads, and one for each remote MPI process.

- Create a directory "lab5". Within this directory, place:
  - all source code files;
  - makefile
    - your program should build with the command “make” with no parameters.
    - name your executable “lab5_mpi.”
- submit your report in .pdf format via Carmen.

MPI on OSC

MVAPICH2, an open source standard MPI implementation, is installed and available on the Owens cluster. Use the *module list* command to verify it is loaded, and *module load mvapich2* if necessary. To allocate a proper node and begin an interactive computing session, use: **qsub -I -l walltime=0:59:00 -l nodes=1:ppn=28** (qsub -"capital eye" -"little ell" walltime . . . -"little ell" nodes . . .). (Note: ppn=28 is appropriate for Owens, use ppn=12 if you are testing on Oakley.)

The qsub command often obtains an interactive session within a few minutes. Note, however, that OSC is a shared resource, and your wait will very with current system load. The load on Owens is anticipated to increase significantly near the end of the semester. Please plan accordingly. We are guests on the OSC cluster, so please practice good citizenship. You can compile your MPI programs on the "login nodes." Only start a qsub interactive session **when you are ready to run your program.** Please "exit" from qsub once your test is complete. If you need to make significant revisions to your source, exit qsub, edit as required, and then start a new qsub session to re-test. Also, do not forget we have access to the debug and batch queues as well.

The MPI compiler is *mpicc*, which uses the same options as the gcc C compiler.

To execute MPI programs on Owens first start an interactive qsub session, and then use *mpirun -np 5 ./lab5_mpi*. The "-np 5" option will specify the 5 processes (an MPI master process and the four remote MPI computational processes) required for this assignment.