CS 241-96
MP 1: Advanced Parallel Programming*

Assigned: March 2, 2016
Due: 11:59PM CST, March 20, 2016

---

1 Running Your Code

In this MP, we will be using software which is not installed on your CS 241 virtual machines. Instead, you will have to work on EWS machines. You may use any of the lab computers in Siebel, DCL, Grainger, ECEB, etc., or work remotely by logging into `linux.ews.illinois.edu` via `ssh`.

The tools we will need are not available to you by default on EWS. Before beginning work on this assignment, add the following lines to the end of your `~/.bashrc`:

```
module load mpich2
module load intel-license/ews
module load intel-composer
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/class/cs232/REU/tbb/lib/
```

To make these changes take effect, either restart your terminal or run

```
$ source ~/.bashrc
```

Failure to do this can result in one or both of the following:

- The timing library yelling at you and refusing to compile.
- MPI yelling at you and refusing to compile *or* refusing to run *or* running but producing incorrect results.

If you choose not to complete this MP on EWS, we ***will not*** help you configure your machine.

2 Problems

1. (30 points) **Getting Started**

   In this part of the MP, you will familiarize yourself with OpenMP and MPI by writing some simple parallel code. Specifically, you will change all the text in a file to upper-case. You can find the serial version of this code in `toupper_serial.c`.

   You can run this code with

   ```
   ./toupper_serial big_text_file.txt
   ```

   If you want to see the output from your code, run

   ```
   ./toupper_serial -print big_text_file.txt
   ```

*Based on exercises written by Craig Zilles

The interface for each parallel version of `toUpper` is the same as the interface for the serial one.

Running the provided makefile will produce `shakespeare.txt`, which is a text file containing the complete text of every work of William Shakespeare. You also have access to `eight_shakespeares.txt`, which is the contents of `shakespeare.txt` repeated eight times. Both of these are good text files to use when benchmarking your parallel code against the serial version.

(a) (10 points) Implement `toUpper` using OpenMP in `toupper_omp.c`.

(b) (20 points) Implement `toUpper` using MPI in `toupper_mpi.c`. You will need to do some extra work to move the transformed text back to a single process in order to print it. We will test your code using the `-print` command line argument, so you must implement printing to receive full credit for this part.

2. (70 points) **Parallel Histograms**

In this part of the MP, you will write code to count the frequency of characters in a text file. You will start by writing serial code for this and then moving on to OpenMP and MPI.

You can run the serial version of this code with

```
./histogram_serial big_text_file.txt
```

If you want to see the character frequencies your code calculates, run

```
./histogram_serial -print big_text_file.txt
```

The interface for each parallel version of `histogram` is the same as the interface for the serial one.

(a) (10 points) Implement the serial version of `histogram` in `histogram_serial.c`.

(b) (30 points) Implement `histogram` using OpenMP in `histogram_omp.c`.

(c) (30 points) Implement `histogram` using MPI in `histogram_mpi.c`.

3 Grading

Your grade will depend on both your code's correctness and its running time. That is, your code should not only be correct, but it should be reasonably faster than the provided serial solution. In general, you should aim to have parallel code that runs in about 2/3 the time the serial solution takes ***when no input or output is taking place***. Note that this is a *general* guideline; achieving this 1.5x speedup is not a guarantee of full points for running time on all problems. There is no penalty for writing correct code which achieves greater speedup.

Your code will be tested on EWS machines; you should do the same when evaluating your code's running time. Do not use a remote connection to evaluate running time, as you may get artificially bad results.

4 Turning in the Assignment

Turn in your code by committing to `svn`. The following files will be graded:

- `toupper_omp.c`
- `toupper_mpi.c`
- `histogram_serial.c`
- `histogram_omp.c`
- `histogram_mpi.c`

In particular, ***do not modify the provided makefile***.