CISC 372 Introduction to Parallel Programming - Fall 2003

The Great Parallel Programming Contest of the Millennium
Contest Rules

Who

This is your last group project in CISC 372. It is not an optional assignment. Contest participants will compete in teams of two to four students. Each team will consist solely of students actively registered for CISC 372-010-F03. **Each team must register their team by sending an email to Lori Pollock at pollock@cis.udel.edu by Monday, December 1, 2003. The email should contain the names of the team members and team name.**

What

The challenge is to write a parallel sort program in C utilizing MPI. Given an input file, your program must create a new file that contains all the input sorted in non-decreasing order. The actual parallel sorting algorithm that you use is up to you.

When

The contest will begin on December 1, 2003 and last through December 9, 2003. All work must be submitted no later than 11:59 PM, December 9, 2003. Late entries will not be accepted.

Submission

Each team will submit its entry by mailing all relevant code, header files, documentation, and makefiles to qili@cis.udel.edu. Include all files as mime attachments (this can easily be done in pine). Your email must be time-stamped (by the receiving eecis mail server) earlier than the cut-off time (the time stamp from the local sending machine will not be considered as it can be spoofed). After you submit your files to the TA, do not make any changes to any of your contest files on porsche. Any claims of lost/delayed mail will be resolved by reviewing your files on porsche. If such a review is needed, files with time-stamps after the deadline will invalidate your entry and a score of zero will be given. Not following the submission guidelines will invalidate your entry. You do not have to submit any timing data. We will be compiling and running your programs to time them.

Resources

All work submitted is to be the result solely of the efforts of the members of the team. Teams are not to share information or resources of any kind with other teams. Teams may consult outside resources (textbooks, papers, and the web) to gain better understanding of algorithms and coding constructs. Teams may not utilize code obtained from outside resources. All code is to be the product of the team (with the exception of pre-compiled libraries that are provided by the system).

Questions

The judges will not invite or accept questions about the problem. Clarification about the problem statement is permitted, however. Any questions regarding problem ambiguity should be addressed to the professor or the TA. Replies to any such questions will be directed to the entire class.

Runtime

Input, output and runtime behavior for the problem will conform to the following guidelines:

- Your program will be run with the following parameters:

  ```
  mpirun -np '#' contest 'infile' 'outfile'
  ```

  where '#' will be an arbitrary number of processors
  contest is the executable
  'infile' is the name of the input data
  'outfile' is the name of your output data
  redirection of stdin/stdout is not allowed (you must explicitly deal with file I/O)

- The input file will consist of one integer (could be positive, negative, zero) per line until EOF. Example input files can be found in pollock/372_f03/public/contest/ on porsche

- Your output file will consist of one integer per line until EOF (same format as the input file).

- Entries that cannot read the input files or fail to save the output files in correct format will be deemed incorrect.

- You will print timing data to standard out which will be calculated as follows:
  - The first executable statements immediately following all variable declarations will be the calls to MPI_Init() and the timing function.
  - The last executable statements will be calls to the timing function, printing your timing data, and then MPI_Finalize(). It will look something like this:

  ```
  main(int argc, char *argv[]) {
    /* variable decl's go here... */
    MPI_Init(&argc, &argv);
    t1 = MPI_Wtime();
    /* your very clever solution goes here... */
    t2 = MPI_Wtime();
    if (node==0) printf(``Time: \n'', t2 - t1);
    MPI_Finalize();
  }
  ```

Winners

The winning team will be determined by averaging the score of the five judges:

Lori Pollock, your professor!
Qi Li, your TA
Antony Danalis, CIS PhD student
Ben Breech, CIS PhD student

The team with the highest average score will be declared the winner. Scoring will be according to the criteria posted below. In the event of a tie, multiple winners will be declared.

Scoring

Your program will be compiled and run with the sample test data and additional test cases created by the judges for the scoring. The following criteria will be used for scoring.

1. Correctness
   - Solution must be computationally correct
2. Performance
   - Execution time on more than one processor will be heavily considered
   - Efficiency of space will also be considered, but not as strongly (as long as your solution is not really offensive)
   - Scalability in terms of input and number of processors
   - Load balancing
3. Creativity/Ingenuity
   - Choice of most appropriate MPI constructs
   - Cleverness of solution
4. Generality
   - Able to accept range of input (in terms of data values and amount of input data)
   - Able to run on an arbitrary number of processors
5. Documentation
   - Internal: code must be well documented
   - External: well organized, well written documentation which addresses:
     - Members of the team
     - Discussion of algorithm used
     - Anything else you feel is important to communicate to the judges

   External documentation can be in ASCII text, postscript, or pdf form only (no Word docs, etc.)

Your submission will be scored within each of the above criteria on a scale from one to five (one being a poor rating, five being an excellent rating). The scores for all the criteria will be averaged to create an overall score for your submission by each judge. The judges' decision is final.

Assessment for Grades

Individual grades for this contest will be assigned based on an evaluation by your professor (using peer evaluation input) separate from the judging for the contest. Grades will be assigned according to the following scale:

- 40% Correctness
- 40% Performance
- 8% Creativity
- 7% Generality
- 5% Documentation