Deadline: 10-11-2017, 23.59.

Homework 1

The purpose of this assignment is to get some hands-on experience with optimizing parallel program performance. Although the task which you are asked to parallelize is relatively simple, there are a number of subtle issues related to achieving high performance.

Description: primes.c is a serial program written in C for determining prime numbers. The program works by testing each odd number (up to a specified limit) for divisibility by all of the factors from 3 to the square root of that number. The algorithm is not very smart, but it is easy to parallelize.

Your assignment is to parallelize this existing algorithm using pthread and OpenMP for two different versions.

The program takes two main parameters, which are read in from the command line:

1-P: the number of processors (numProcs in the code); and
2-N: the problem size (size in the code).

In addition, the program takes arguments that output all of the primes generated, either to a file or standard out. This should assist you in ensuring that the parallel version of the algorithm works correctly. Precise usage instructions can be seen by running ./primes –h and you can make use of the given Makefile.

The main data structure of the program is an array which holds boolean values indicating whether the corresponding numbers are prime. The array only holds odd numbers, since no even numbers (except 2) are prime. You should parallelize the actual core code.

We would like for you to implement two different versions of the parallel code and compare their performance:

1- Pthread version
2- OpenMP version

One trick is to achieve good "load balancing" with minimal overhead while achieving good memory performance. You should be able to achieve very good speedups on this assignment (i.e. close to linear).

To evaluate the performance of the parallel program, measure the following times using gettimeofday function also given in the code as gettime(). Use only computation time for timing; this is strictly the time to compute the answer. Start timing when the main computation starts (after all the processes have been created), and finish when all of the answers have been calculated. Additionally, when timing the program DO NOT print all the prime numbers on the screen.

Tests--------

Do tests for these settings:

1- Run Pthread and OpenMP versions for N value 200K , 500K, 1M, 2M, 4M and P=1,2,4,8
2- Give running times in a table for Pthread and OpenMP

3- For the fastest thread configuration of each N give speed up values for Pthread and OpenMP.

Notes

- What to submit:
  - Source codes with Makefile
  - README file (preferably in pdf)
    - table of performance results for
      - Question 2
      - Question 3
  - compress all as a tarball
- Turn in the compressed file via **submit system.**
- Don’t forget to comment your code for readability. Code without any documentation will lose 5 points
- This is an individual homework
- dev.cs.hacettepe.edu.tr machines are ready for OpenMP and Pthread development