Assignment #1: Programming with Pthread and OpenMP (Due data: 11:59pm April 21st)

Solving a linear system in parallel using Gaussian elimination

Gaussian elimination is a classical method for solving a matrix of linear equations of the form Ax = b. By performing elementary row operations, Gaussian elimination transforms the square matrix A into an equivalent upper-triangular matrix. Following transformation of A into upper triangular form, a back substitution phase solves for x. A high level overview of Gaussian elimination and back substitution can be found on Mathworld.

In this assignment, you will develop a parallel linear solver that uses Gaussian elimination with partial pivoting (partial pivoting is used to enhance numerical stability) to transform a dense n x n matrix into an upper-triangular one and then solve the resulting linear system by back substitution. Below is pseudocode written using Fortran 90 array notation for a sequential implementation of Gaussian elimination with partial pivoting.

```
inputs: a(n,n), b(n)
outputs: a(n,n), b(n) in echelon form

do j=1,n-1
  ksave = maxloc(abs(a(j:n,j)))
  k = ksave(1) + j-1
  swap a(j,:) and a(k,:)
  swap b(j) and b(k)
  do k=j+1, n
    m = a(k,j)/a(j,j)
    a(k,j:) = a(k,j:) - m*a(j,j:)
    b(k) = b(k) - m*b(j)
  enddo
enddo
```

You will use the pthread and openMP shared-memory programming model to write a parallel linear solver based on Gaussiam elimination with partial pivoting. The parallel solver implementation should accept two arguments: n - the size of a matrix, followed by p - the number of threads. Your programs will allocate an n x n matrix a and an n-vector b of double precision (64-bit) floating point variables; both should be filled with random floating point numbers whose values are based on the drand48 or drand48_r random number generators. See the man pages for details. (Note: if you are generating random numbers in parallel, you will have to use a reentrant random number generation routine and seed the random number generators for each thread differently.) Apply Gaussian elimination with partial pivoting to transform the matrix into an upper triangular one, and then apply back substitution to compute x. To check your answer, compute the square root of the sum of the squares of the residual vector (this sum is known as the L2-norm) computed as Ax-b. Print the value of the L2-norm of the residual. (It should be very small.)

The verification step need not be parallelized. Have your program time the Gaussian elimination followed by back substitution phase by reading the real-time clock before and afterward and printing the difference.

The formal components of the assignment are listed below:

- Write a shared-memory parallel program that uses Pthreads to solve a linear system using Gaussian elimination with partial pivoting.
- Write a shared-memory parallel program that uses OpenMP to solve a linear system using Gaussian elimination with partial pivoting.

- Write a document that describes how your programs work. This document should not include your programs, though it may include figures containing pseudo-code that sketch the key elements of your parallelization strategy for each implementation. Explain how your program partitions the data, work and exploits parallelism. Justify your implementation choices. Explain how the parallel work is synchronized.

- Your report may include performance comparison between the openmp and pthread implementations. Also, you can compare them with respect to their programmability.

Use problem size n = 8000 to evaluate the performance of your implementations. If your sequential running time is too long for the interactive queue, you may base your timing measurements on n=6000 or n=5000. Prepare a table that includes your timing measurements for the combination of the elimination and back substitution phases of your implementations on 1-12 cores on a node on the cacloud servers. Graph of the parallel efficiency of your program executions. Plot a point for each of the executions. The x axis should show the number of processors. The Y axis should show your measured parallel efficiency for the execution. Construct your plot so that the X axis of the graph intersects the Y axis at Y=0.

Your score is not only dependent upon the speedup, but also upon the quality of your writeup.

Your submission should contain:

- The code for your pthread and openmp program and a Makefile to build the code, and

- A writeup about your programs in PDF format.

Send your code and writeup to bokyeong@camars.kaist.ac.kr

This assignment is adapted from "COMP 422 Parallel Computing" at Rice University