CS69001 Computing Laboratory – I

Assignment No: C4

Date: 05–November–2018

Make a parallel implementation of the standard linear-system solver algorithm (Gaussian elimination) using POSIX threads. Consider a square system of linear equations

$$A\mathbf{x} = \mathbf{b},$$

where A is an $N \times N$ matrix, and $\mathbf{x}$ and $\mathbf{b}$ are N-dimensional (column) vectors, all with real-valued (floating-point) elements. Assume that the system is uniquely solvable (for $\mathbf{x}$). Take $N = 1000$ for this assignment.

Gaussian elimination

Let a_{ij} denote the (i,j)-th entry of A, and b_i, x_i the i-th entries of $\mathbf{b}$ and $\mathbf{x}$, respectively. Let us also use the symbol A_i to denote the i-th row of A. Array indexing is assumed to be zero-based.

A. Triangularize the matrix A in place

For $i = 0, 1, 2, \ldots, N-1$, repeat Steps 1–4:

1. Try to locate a $k \in \{i, i+1, i+2, \ldots, N-1\}$ such that $a_{ki} \neq 0$.
   If no such k is found, the system is underspecified, exit.
2. If $k > i$, swap A_i with A_k, and b_i with b_k.
3. Divide A_i and b_i by a_{ii} (so a_{ii} becomes 1 after this).
4. For $k = i+1, i+2, \ldots, N-1$, repeat:
   If $a_{ki} \neq 0$, subtract a_{ki} times A_i from A_k, and a_{ki} times b_i from b_k.

B. Backward substitution

5. For $i = N-1, N-2, \ldots, 2, 1, 0$, compute the solution $x_i = b_i - \sum_{j=i+1}^{N-1} a_{ij} x_j$.

The threaded implementation

The master thread M and p worker threads $W_0, W_1, W_2, \ldots, W_{p-1}$ run this algorithm as follows. The number p of worker threads is to be read from the user.

The work of M	**The work of W_t, $t \in \{0, 1, 2, \ldots, p-1\}$**
Initialize the system of equations. Create and initialize mutexes and condition variables. Create the worker threads $W_0, W_1, W_2, \ldots, W_{p-1}$. Wait for a while to allow the worker threads to start.	
For $i = 0, 1, 2, \ldots, N-1$, repeat:	For $i = 0, 1, 2, \ldots, N-1$, repeat:
Carry out Steps 1, 2, 3. Break the interval $[i+1, N-1]$ into (nearly) equal-sized sub-intervals $I_0, I_1, I_2, \ldots, I_{p-1}$. Signal all the worker threads. Wait until all worker threads are done.	Wait until signaled by M. Carry out Step 4 for all $k \in I_t$. If W_t is the last thread to finish the current iteration, wake up the master thread M.
	Terminate.
Wait until all W_t, $t \in \{0, 1, 2, \ldots, p-1\}$, terminate. Carry out Step 5. Verify the correctness of the solution.	

Notice that no iteration of the row-reduction loop may start before the previous iteration is over. Moreover, the main thread M must do some initial work before the worker threads can start the reduction process. Implement these two instances of synchronization using condition variables.

Remark: Arguably, a more natural way to implement the synchronization of the row-reduction loop is by using barriers. This technique is not covered in the class, so you use condition variables only.

Checking for correctness

The linear system (A and $\mathbf{b}$) can be randomly generated as follows. Choose each

$$a_{ij} \in_R \{-9,-8,\ldots,-2,-1,0,1,2,\ldots,8,9\}.$$

Also, generate a solution $\mathbf{y}$ with each entry chosen randomly in the same range. Compute $\mathbf{b} = A\mathbf{y}$. Forget $\mathbf{y}$. This ensures that the system is solvable. Its unique solvability (that is, $\mathrm{Rank}(A) = N$) cannot be guaranteed by this random generation process, but the situation $\mathrm{Rank}(A) < N$ occurs with very low probability. The master thread should keep a copy A' of A, and a copy $\mathbf{b}'$ of $\mathbf{b}$.

Although the elements of $A, \mathbf{b}, \mathbf{y}$ are chosen as integers, we need these elements to be floating-point variables (preferably `double`). Gaussian elimination cannot proceed with integer arithmetic.

After the master thread computes $\mathbf{x}$, it needs to check whether $A'\mathbf{x} = \mathbf{b}'$ (initial system) and $A\mathbf{x} = \mathbf{b}$ (reduced system). However, the use of floating-point arithmetic implies introduction of approximation errors. That is, the above equalities are only approximately satisfied. Compute and report the approximation errors as the largest entries of $A'\mathbf{x} - \mathbf{b}'$ and $A\mathbf{x} - \mathbf{b}$. These errors should be rather small.

Sample output

```
+++ p = 8

+++ Worker thread 1 starting
+++ Worker thread 0 starting
+++ Worker thread 3 starting
+++ Worker thread 4 starting
+++ Worker thread 2 starting
+++ Worker thread 5 starting
+++ Worker thread 6 starting
+++ Worker thread 7 starting

+++ Worker thread 5 exiting
+++ Worker thread 1 exiting
+++ Worker thread 7 exiting
+++ Worker thread 4 exiting
+++ Worker thread 2 exiting
+++ Worker thread 6 exiting
+++ Worker thread 0 exiting
+++ Worker thread 3 exiting

+++ Master thread: Row reduction done

+++ Master thread: Solution computed
    Maximum error (initial system) = 0.00000016573313
    Maximum error (reduced system) = 0.00000000013097
```

Submit a single C/C++ source file. Use no non-shared global variables.