CS451 “Introduction to Parallel and Distributing Processing”

Homework 2 report

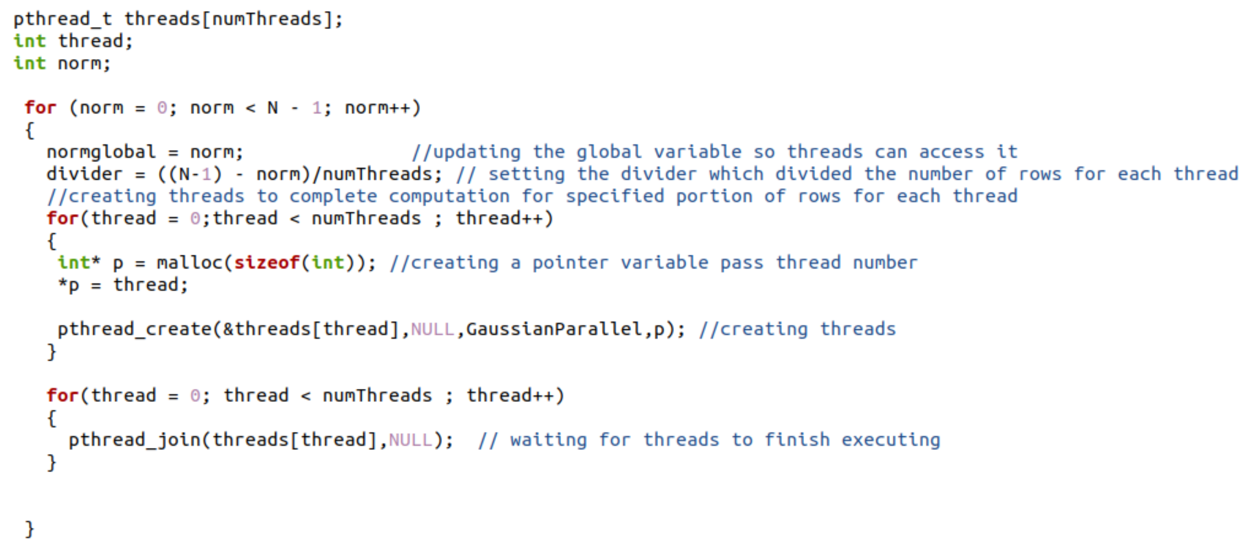
Jayanth Chidananda
A20517012

Pthreads

Techniques used to optimize the gaussian elimination in **pthreads**.

1. The inner two for loops which find the multiplier for the row below the diagonal and perform multiplication on all column elements. The technique I've followed creates threads to process all the rows below the diagonal in one iteration, and join them. And repeating the process again until all the subdiagonal elements are zero.

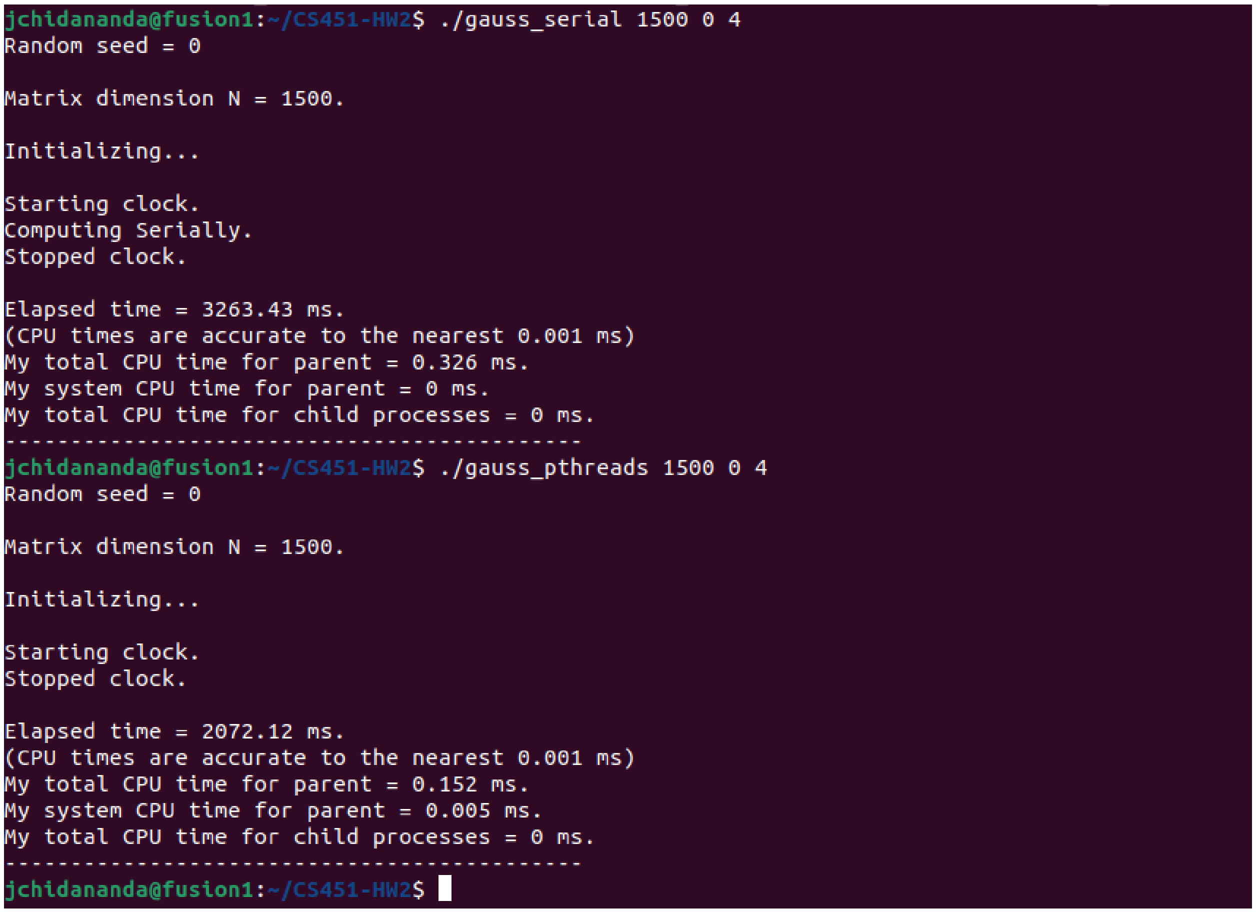
2. A global variable named divider sets the number of rows each thread should process before finishing execution.



```
pthread_t threads[numThreads];
int thread;
int norm;

 for (norm = 0; norm < N - 1; norm++)
 {
   normglobal = norm;               //updating the global variable so threads can access it
   divider = ((N-1) - norm)/numThreads; // setting the divider which divided the number of rows for each thread
   //creating threads to complete computation for specified portion of rows for each thread
   for(thread = 0;thread < numThreads ; thread++)
   {
    int* p = malloc(sizeof(int)); //creating a pointer variable pass thread number
    *p = thread;

    pthread_create(&threads[thread],NULL,GaussianParallel,p); //creating threads
   }

   for(thread = 0; thread < numThreads ; thread++)
   {
     pthread_join(threads[thread],NULL);  // waiting for threads to finish executing
   }


 }
```

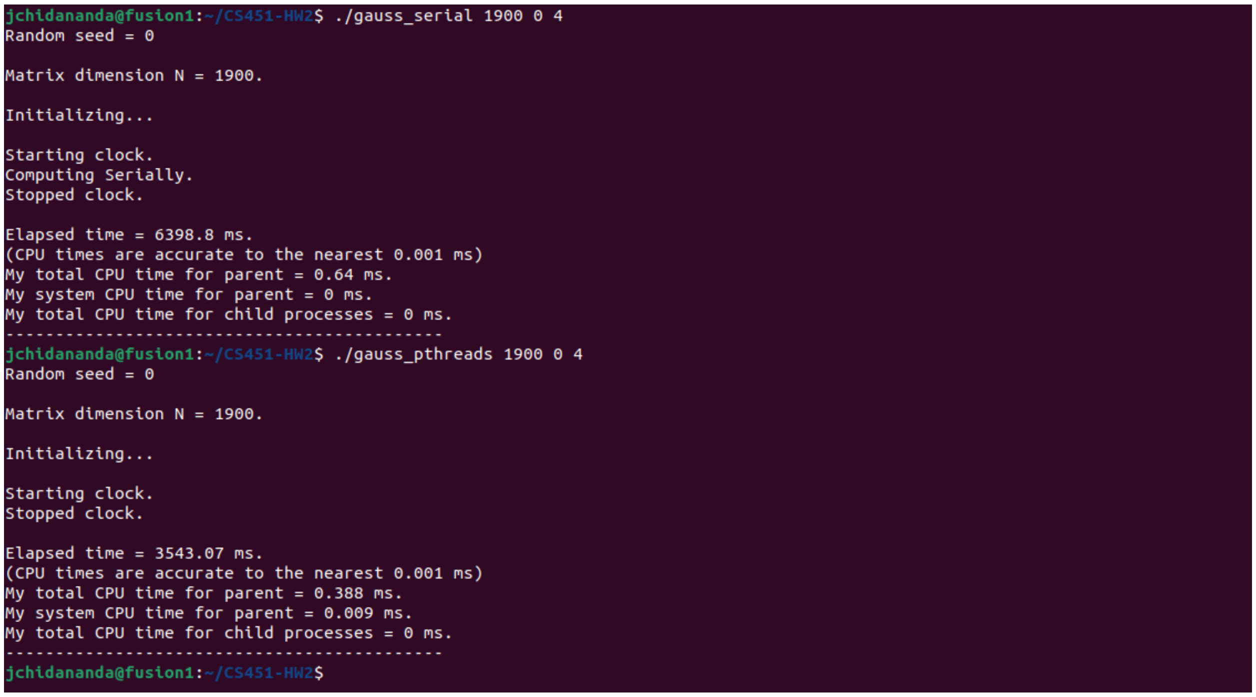
Performance results

Eg.
1) For a matrix of size 1500 and number of threads set to 4, the serial execution time was 3263.43 ms, with CPU time being 0.326 ms. while, with pthreads for the same size of 1500 the execution time was 2072.12 ms, with CPU time being 0.152 ms. reducing the execution time by more than a sec, and also increasing the CPU time, increasing CPU utilization as well.



```
jchidananda@fusion1:~/CS451-HW2$ ./gauss_serial 1500 0 4
Random seed = 0

Matrix dimension N = 1500.

Initializing...

Starting clock.
Computing Serially.
Stopped clock.

Elapsed time = 3263.43 ms.
(CPU times are accurate to the nearest 0.001 ms)
My total CPU time for parent = 0.326 ms.
My system CPU time for parent = 0 ms.
My total CPU time for child processes = 0 ms.
--------------------------------------------
jchidananda@fusion1:~/CS451-HW2$ ./gauss_pthreads 1500 0 4
Random seed = 0

Matrix dimension N = 1500.

Initializing...

Starting clock.
Stopped clock.

Elapsed time = 2072.12 ms.
(CPU times are accurate to the nearest 0.001 ms)
My total CPU time for parent = 0.152 ms.
My system CPU time for parent = 0.005 ms.
My total CPU time for child processes = 0 ms.
--------------------------------------------
jchidananda@fusion1:~/CS451-HW2$
```

2) For a matrix of size 1900 and number of threads set to 4, the serial execution time was 6398.8 ms, with CPU time being 0.64 ms. while, with pthreads for the same size of 1500 the execution time was 3543.07 ms, with CPU time being 0.388 ms. reducing the execution time to nearly half of the serial execution time , and also increasing the CPU time, increasing CPU utilization as well.



```
jchidananda@fusion1:~/CS451-HW2$ ./gauss_serial 1900 0 4
Random seed = 0

Matrix dimension N = 1900.

Initializing...

Starting clock.
Computing Serially.
Stopped clock.

Elapsed time = 6398.8 ms.
(CPU times are accurate to the nearest 0.001 ms)
My total CPU time for parent = 0.64 ms.
My system CPU time for parent = 0 ms.
My total CPU time for child processes = 0 ms.
--------------------------------------------
jchidananda@fusion1:~/CS451-HW2$ ./gauss_pthreads 1900 0 4
Random seed = 0

Matrix dimension N = 1900.

Initializing...

Starting clock.
Stopped clock.

Elapsed time = 3543.07 ms.
(CPU times are accurate to the nearest 0.001 ms)
My total CPU time for parent = 0.388 ms.
My system CPU time for parent = 0.009 ms.
My total CPU time for child processes = 0 ms.
--------------------------------------------
jchidananda@fusion1:~/CS451-HW2$
```

Conclusion: we can see the program has considerable performance improvement for larger matrix sizes. With the program being effectively parallelized.

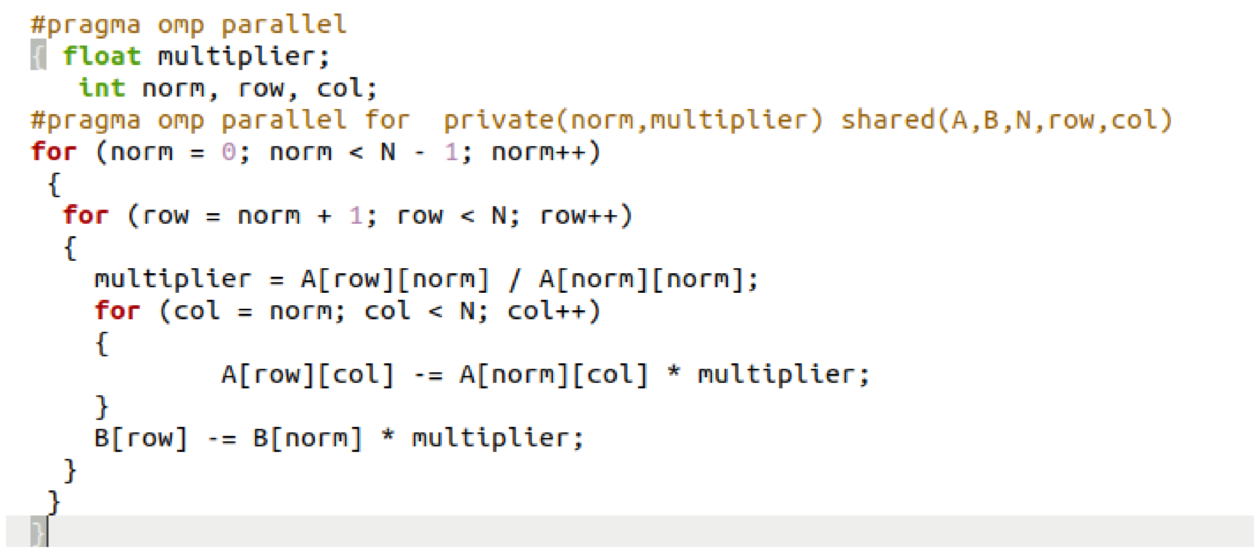
Other versions

Parallelizing the first outer for loop, leads to race conditions as one iteration changes almost the entire matrix, preventing other threads from accessing the correct result and producing incorrect results.

OpenMP

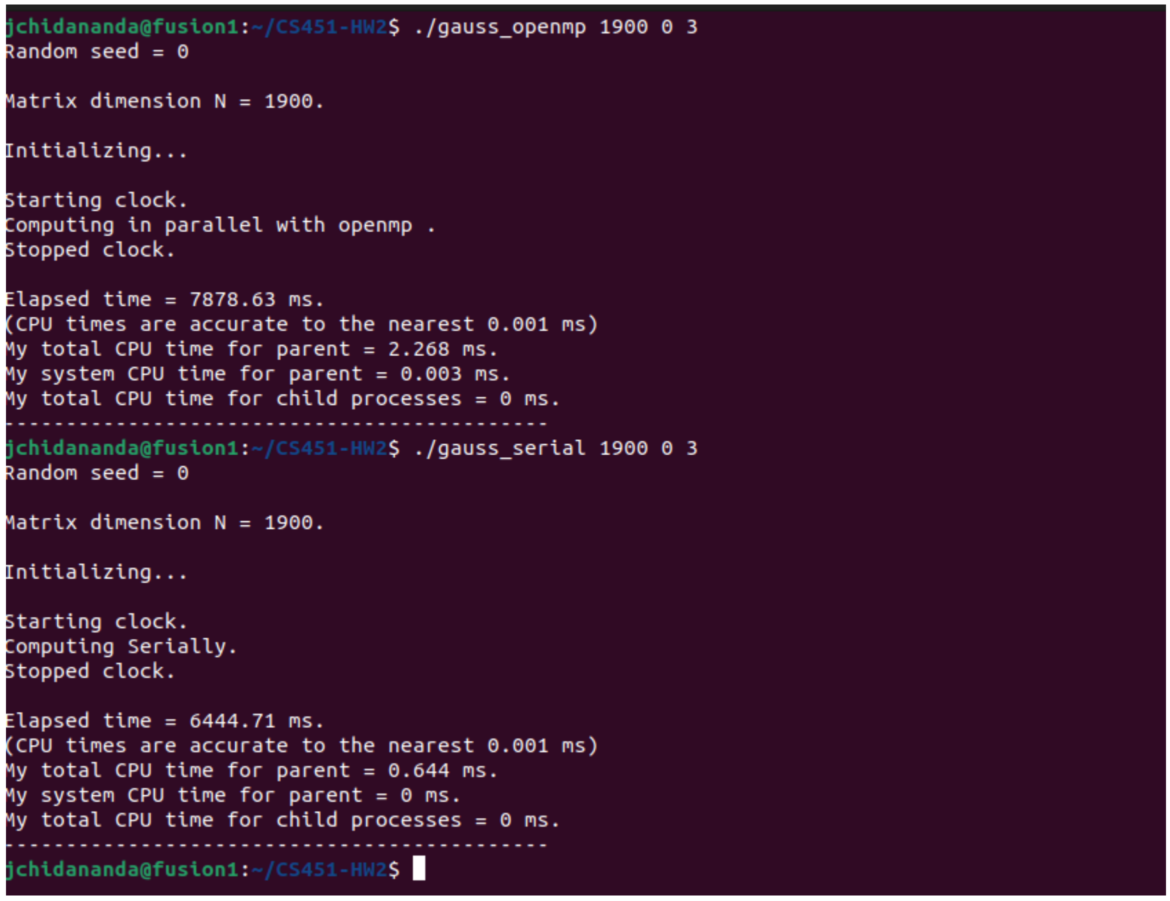
Techniques used to optimize the gaussian elimination in openmp.

1. With openmp, the entire for loop was placed inside the pragma omp parallel directive region, and the pragma omp for loop directive was placed just above the outer for loop, and the clauses which specified the private and shared variables.



```
#pragma omp parallel
{ float multiplier;
   int norm, row, col;
#pragma omp parallel for  private(norm,multiplier) shared(A,B,N,row,col)
for (norm = 0; norm < N - 1; norm++)
 {
  for (row = norm + 1; row < N; row++)
  {
    multiplier = A[row][norm] / A[norm][norm];
    for (col = norm; col < N; col++)
    {
            A[row][col] -= A[norm][col] * multiplier;
    }
    B[row] -= B[norm] * multiplier;
  }
 }
}
```

Performance results: 1) With the size of the being set to 1900, the serial execution time was 6444.71 ms, while the openmp with 3 threads, the execution time was 7878.63 ms, although there is an increase in execution time , it is largely due to the overhead of the threads.



```
jchidananda@fusion1:~/CS451-HW2$ ./gauss_openmp 1900 0 3
Random seed = 0

Matrix dimension N = 1900.

Initializing...

Starting clock.
Computing in parallel with openmp .
Stopped clock.

Elapsed time = 7878.63 ms.
(CPU times are accurate to the nearest 0.001 ms)
My total CPU time for parent = 2.268 ms.
My system CPU time for parent = 0.003 ms.
My total CPU time for child processes = 0 ms.
--------------------------------------------
jchidananda@fusion1:~/CS451-HW2$ ./gauss_serial 1900 0 3
Random seed = 0

Matrix dimension N = 1900.

Initializing...

Starting clock.
Computing Serially.
Stopped clock.

Elapsed time = 6444.71 ms.
(CPU times are accurate to the nearest 0.001 ms)
My total CPU time for parent = 0.644 ms.
My system CPU time for parent = 0 ms.
My total CPU time for child processes = 0 ms.
--------------------------------------------
jchidananda@fusion1:~/CS451-HW2$ 
```

Conclusion: Although we can see the program execution time, not being considerably reduced, we can not conclude that it is due to bad parallelization, as most of it would have been due to overhead associated with threads and false sharing, where each thread may fetch the block again even though it has not been modified.