Report
Assignment 2

Krishnakant Dharekar 2018CS10351

I will try to explain each method seprately

1) method 1

Approach : since we can see that for calculating the value of L[i][j] it is dependent on the previous values of L on the same row for row = i and value of column is from 0 to j and on the values of U where it is dpendent on the values of U for column is j and row is <= j .
so it can be seen that we cannot parallelize the outer loop directly as this would voilate the loop carried dependency that is existing in the loop.
I have parallelized the inner loop which has i as the iterator . Since it doesnot use value of some location of L or U which is to be changed in the future iterations.

In this part the sum variable has data race so I have avoided it by making it private variable .

2) method 2

Approach : since we can see that for calculating the value of L[i][j] it is dependent on the previous values of L on the same row for row = i and value of column is from 0 to j and on the values of U where it is dependent on the values of U for column is j and row is <= j .

I have applied sections construct twice in this . What I have done is broken both the for loops seprately into 16 sections to be devided among the the available number of threads . Since the maximum possible number of threads that we can have is 16 so we are able to devide each section between the threads no matter how many threads we have.

Sine in this implementation I have devided the loop into chunks and done the calculation. So there is no possibility of data race .

3) method 3

Approach : since we can see that for calculating the value of L[i][j] it is dependent on the previous values of L on the same row for row = i and value of column is from 0 to

j and on the values of U where it is dependent on the values of U for column is j and row is <= j .

In this method what I have done is I tried to mearge both my approches taken in method1 and method2. First I applied parallel for construct on the for loops with iterator i and then I broke the ineer loop which as iterator K into two sections. The threads are also devided like this that the for loop has total_num_threads/2 and the inner loop has 2 threads .

I made the variables sum1, sum2 , sum3 , sum4 as private in place of a single sum and made separate sum variables for these for every internal section so that there is no data race for accessing the sum value.

4) method 4:

Here I implemented the mpi version of the code. The approach takin can be explained like I have kind of parallelized the loop with i as its iterator . And I have given the processors the iterations based on the mod value obtaind by dividing i with the total number of processes. And when a process finishes its calculation of some term of L OR U , I broadcast this value to other processes who store this value on their separate L and U matrices.

I have avoided any potential data races in my code using the MPI_Barrier method of mpi so that in case some process finishes its assigned loop iteration then it will have to wait for other processes to finish before it can move on to the next block of code. It has helped in avoiding a data race condition since it could have been possible that some process would have asked for a value of L OR U which was not yet computed by the assigned process and which would then result in wrong answer.