PARALLEL

DISTRIBUTED AND COMPUTING

NAME:YASHRAJ AGARWAL
REGISTRATION NUMBER:18BCI0183

LU DECOMPOSITION USING OPENMP

Task parallelism is a popular and relatively new approach to exploiting parallelism because it offers the potential for minimal synchronization.Tasks were introduced in OpenMP version 3.0 but there are not many instances of them being used in current programming applications. Many scientific applications have linearsystems Ax=b which needs to be solved for different vectors b. Gaussian elimination, or otherwise known as LU decomposition, is an efficient technique to solve a linear system.The main idea of the LU decomposition is to factorize A into an upper (U) and a lower(L) matrix such that A =LU.This ppt presents an OpenMP task parallel approach for the LU factorization.The tasking model is based on the individual computational tasks which occur during the block-wise LU factorization.The algorithm is especially suited for multi core processors and shows a much improved parallel scaling behavior compared to a naive parallel for based LU decomposition

CODE

```cpp
#include <bits/stdc++.h>
#include<omp.h>
using namespace std;

const int MAX = 100;

void luDecomposition(int mat[][MAX], int n)
{
    int lower[n][n], upper[n][n];
    memset(lower, 0, sizeof(lower));
    memset(upper, 0, sizeof(upper));
    int i,j,k;
    for ( i = 0; i < n; i++) {
        #pragma omp parallel for shared(mat,n,i) private(k,j) schedule(static,64)
        for ( k = i; k < n; k++) {
            int sum = 0;
            for ( j = 0; j < i; j++)
                sum += (lower[i][j] * upper[j][k]);
            upper[i][k] = mat[i][k] - sum;
        }
        for ( k = i; k < n; k++) {
            if (i == k)
                lower[i][i] = 1; // Diagonal as 1
            else {
                int sum = 0;
                for ( j = 0; j < i; j++)
                    sum += (lower[k][j] * upper[j][i]);
                lower[k][i] = (mat[k][i] - sum) / upper[i][i];
            }
        }
    }
    cout << setw(6) << "Lower Triangular"<< setw(32) << "Upper Triangular" << endl;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)
            cout << setw(6) << lower[i][j] << "\t";
        cout << "\t";
        for (int j = 0; j < n; j++)
            cout << setw(6) << upper[i][j] << "\t";
        cout << endl;
    }
}

int main()
```

```
int main()
{
    int mat[][MAX] = { { 3, -1, -3 }, { -3, 5, 4 }, { -5, 1, -6 } };
    luDecomposition(mat, 3);
    return 0;
}
```

OUTPUT