CSE4001 - Parallel and Distributed Computing

Lab 21+22

Digital Assignment- 3

Submitted by: Alokam Nikhitha

Reg No:19BCE2555

QUESTION:

Write a C program to perform parallel matrix multiplication using OpenMP. You should first create three matrices A, B, and C then initialize A and B to some values of your choice. In your code, try to improve the performance by (re)using the same set of threads for initializing A and B and for calculating C.

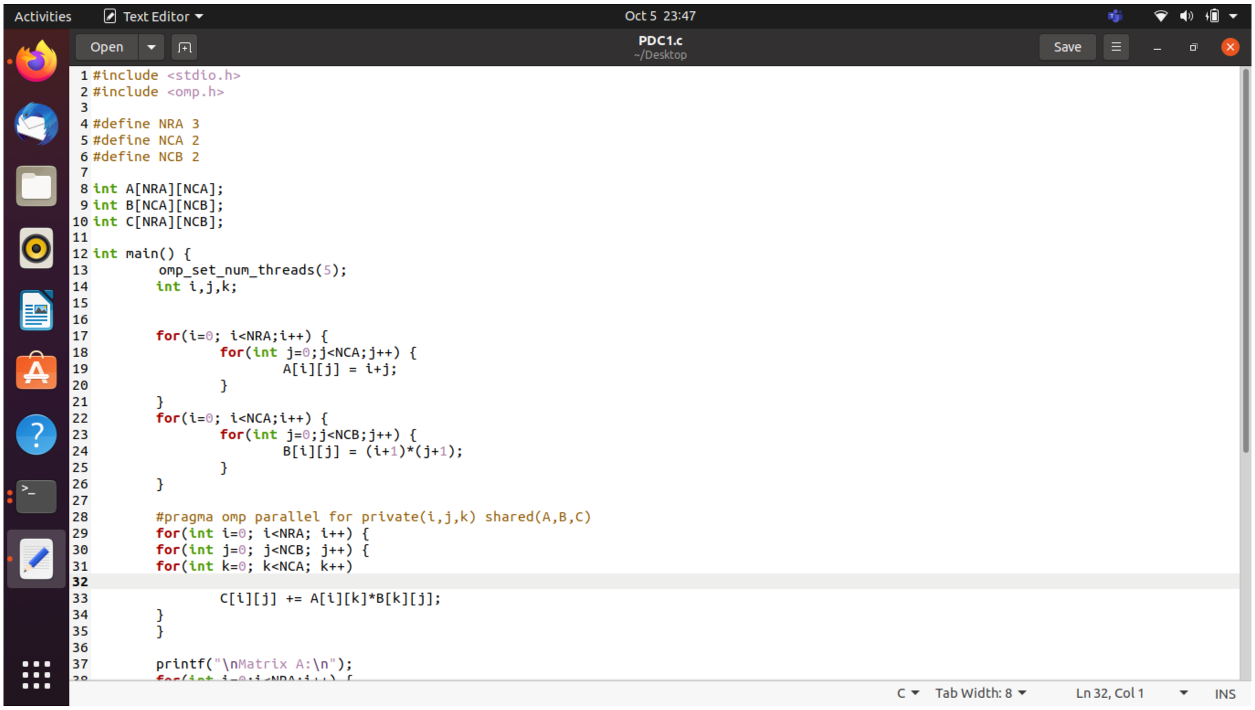
CODE:

```
#include <stdio.h>
#include <omp.h>


#define NRA 3
#define NCA 2
#define NCB 2


int A[NRA][NCA];
int B[NCA][NCB];
int C[NRA][NCB];


int main() {
    omp_set_num_threads(5);
     int i,j,k;
     for(i=0; i<NRA;i++) {
          for(int j=0;j<NCA;j++) {
               A[i][j] = i+j;
          }
     }
     for(i=0; i<NCA;i++) {
```

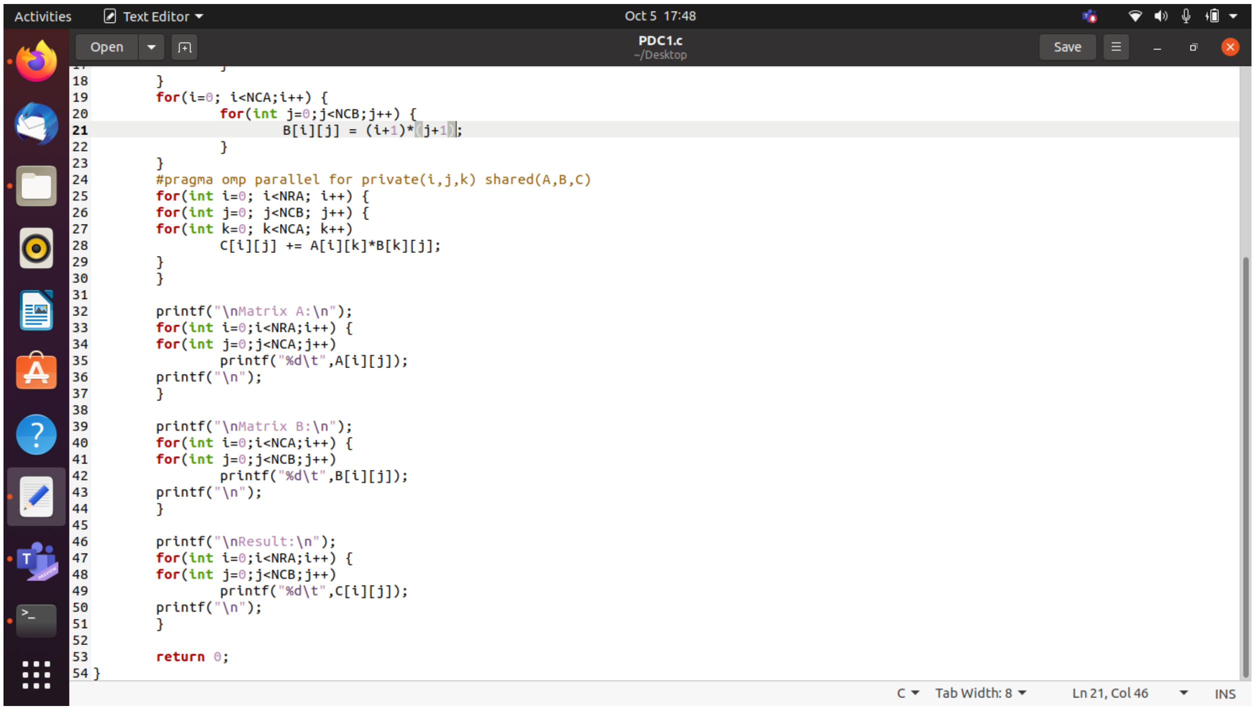
```
    for(int j=0;j<NCB;j++) {
        B[i][j] = (i+1)*(j+1);
    }
}
#pragma omp parallel for private(i,j,k) shared(A,B,C)
for(int i=0; i<NRA; i++) {
for(int j=0; j<NCB; j++) {
for(int k=0; k<NCA; k++)
    C[i][j] += A[i][k]*B[k][j];
}
}

printf("\nMatrix A:\n");
for(int i=0;i<NRA;i++) {
for(int j=0;j<NCA;j++)
    printf("%d\t",A[i][j]);
printf("\n");
}

printf("\nMatrix B:\n");
for(int i=0;i<NCA;i++) {
for(int j=0;j<NCB;j++)
    printf("%d\t",B[i][j]);
printf("\n");
}
```

```
    printf("\nResult:\n");
    for(int i=0;i<NRA;i++) {
    for(int j=0;j<NCB;j++)
            printf("%d\t",C[i][j]);
    printf("\n");
    }


    return 0;
}
```

CODE SNIPPETS:

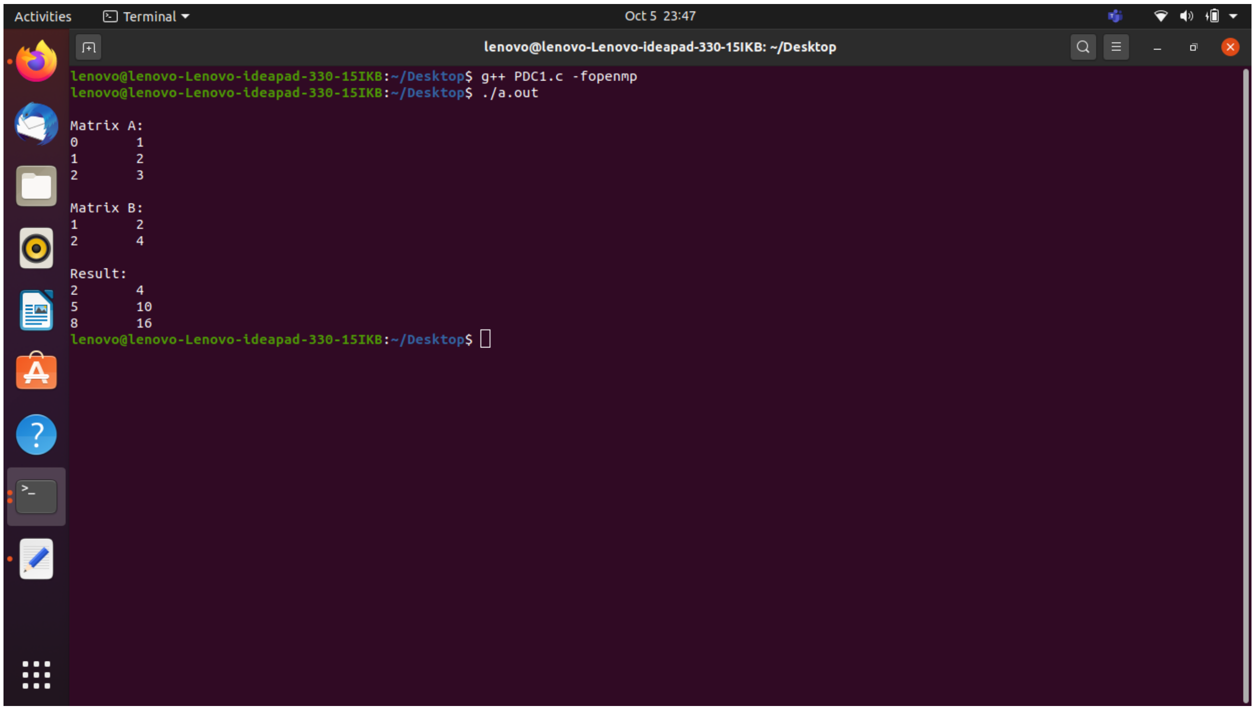


```
#include <stdio.h>
#include <omp.h>

#define NRA 3
#define NCA 2
#define NCB 2

int A[NRA][NCA];
int B[NCA][NCB];
int C[NRA][NCB];

int main() {
         omp_set_num_threads(5);
        int i,j,k;


        for(i=0; i<NRA;i++) {
                for(int j=0;j<NCA;j++) {
                        A[i][j] = i+j;
                }
        }
        for(i=0; i<NCA;i++) {
                for(int j=0;j<NCB;j++) {
                        B[i][j] = (i+1)*(j+1);
                }
        }

        #pragma omp parallel for private(i,j,k) shared(A,B,C)
        for(int i=0; i<NRA; i++) {
        for(int j=0; j<NCB; j++) {
        for(int k=0; k<NCA; k++)

                C[i][j] += A[i][k]*B[k][j];
        }
        }

        printf("\nMatrix A:\n");
        for(int i=0;i<NRA;i++) {
```



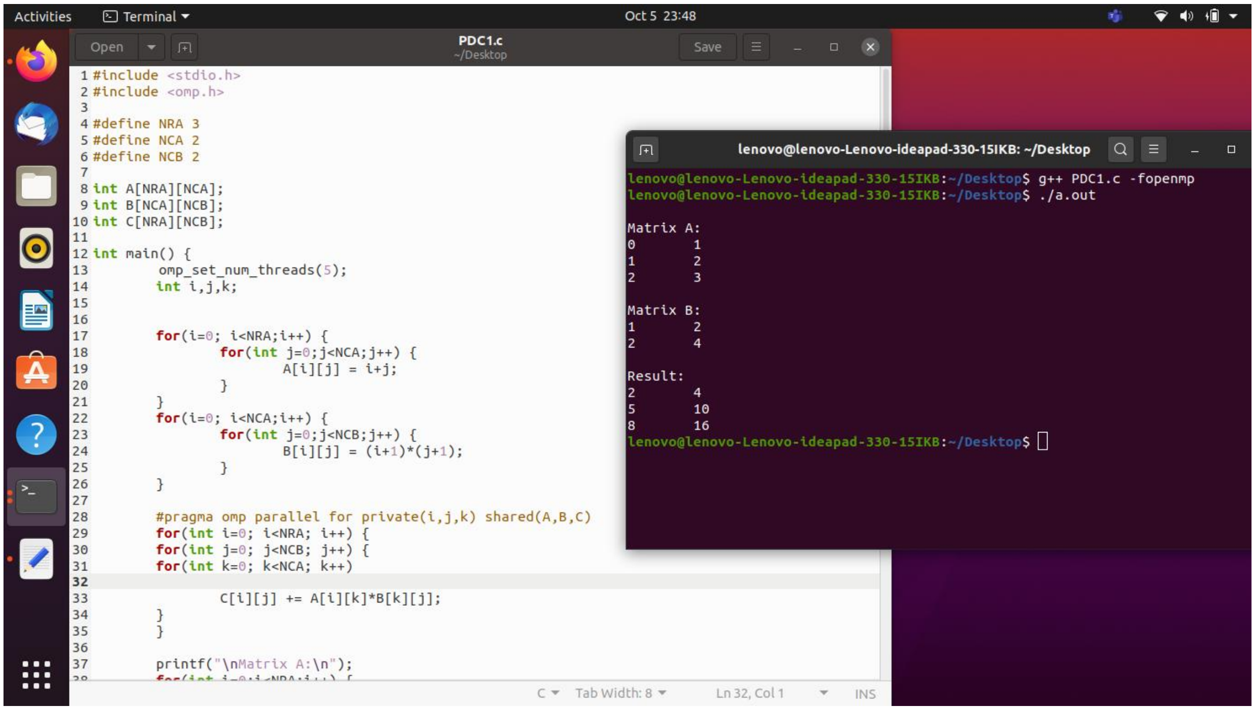
```c
                }
        }
        for(i=0; i<NCA;i++) {
                for(int j=0;j<NCB;j++) {
                        B[i][j] = (i+1)*(j+1);
                }
        }
        #pragma omp parallel for private(i,j,k) shared(A,B,C)
        for(int i=0; i<NRA; i++) {
        for(int j=0; j<NCB; j++) {
        for(int k=0; k<NCA; k++)
                C[i][j] += A[i][k]*B[k][j];
        }
        }

        printf("\nMatrix A:\n");
        for(int i=0;i<NRA;i++) {
        for(int j=0;j<NCA;j++)
                printf("%d\t",A[i][j]);
        printf("\n");
        }

        printf("\nMatrix B:\n");
        for(int i=0;i<NCA;i++) {
        for(int j=0;j<NCB;j++)
                printf("%d\t",B[i][j]);
        printf("\n");
        }

        printf("\nResult:\n");
        for(int i=0;i<NRA;i++) {
        for(int j=0;j<NCB;j++)
                printf("%d\t",C[i][j]);
        printf("\n");
        }

        return 0;
}
```

OUTPUT:



```
lenovo@lenovo-Lenovo-ideapad-330-15IKB:~/Desktop$ g++ PDC1.c -fopenmp
lenovo@lenovo-Lenovo-ideapad-330-15IKB:~/Desktop$ ./a.out

Matrix A:
0       1
1       2
2       3

Matrix B:
1       2
2       4

Result:
2       4
5       10
8       16
lenovo@lenovo-Lenovo-ideapad-330-15IKB:~/Desktop$
```

OUTPUT WITH CODE:



```c
#include <stdio.h>
#include <omp.h>

#define NRA 3
#define NCA 2
#define NCB 2

int A[NRA][NCA];
int B[NCA][NCB];
int C[NRA][NCB];

int main() {
         omp_set_num_threads(5);
        int i,j,k;


        for(i=0; i<NRA;i++) {
                for(int j=0;j<NCA;j++) {
                        A[i][j] = i+j;
                }
        }
        for(i=0; i<NCA;i++) {
                for(int j=0;j<NCB;j++) {
                        B[i][j] = (i+1)*(j+1);
                }
        }

        #pragma omp parallel for private(i,j,k) shared(A,B,C)
        for(int i=0; i<NRA; i++) {
        for(int j=0; j<NCB; j++) {
        for(int k=0; k<NCA; k++)

                C[i][j] += A[i][k]*B[k][j];
        }
        }

        printf("\nMatrix A:\n");
        for(int i=0;i<NRA;i++) {
```

```
lenovo@lenovo-Lenovo-ideapad-330-15IKB:~/Desktop$ g++ PDC1.c -fopenmp
lenovo@lenovo-Lenovo-ideapad-330-15IKB:~/Desktop$ ./a.out

Matrix A:
0       1
1       2
2       3

Matrix B:
1       2
2       4

Result:
2       4
5       10
8       16
lenovo@lenovo-Lenovo-ideapad-330-15IKB:~/Desktop$ 
```