Name: PRIYAL AGRAWAL

Roll No.: 27

Batch: H2

Parallel Programming

Assignment no: 1

Problem Statement: Write a parallel program for matrix multiplication. Measure the performance of the system on the parameters of parallel system performance metrics.

Aim: To write a C program to multiply two large matrix and run the program on a multi-core parallel system.

Objective: To understand the efficiency of a parallel code for matrix multiplication using OpenMP. Understand the concept of granularity and its effects on speedup and efficiency.

Theory:

Algorithm selected for serial to parallel conversion.

Step 1: Divide the both given matrices into 4 equal part

Step 2: Make a pair of each part and assigned to a unique pair

Step 3: Each threads perform multiplication and store the result

Step 4: Do left shift of first matrix and upshift of second matrix

Step 5: Perform Multiplication again and store the second result

Step 6: Add both result will give result

OpenMP construct used:

- OpenMP is an Application Program Interface (API) that may be used to explicitly direct multi-threaded, shared memory parallelism.

```
#pragma omp parallel for private(i, j, k)
```

- Causes the work done in a for loop inside a parallel region to be divided among threads. The private clause allow each thread to have a,b,c as local variables i.e., have different address space

Total Cost = Time complexity × Number of processors used
= O (n^3) × 4
= 4n^3

Speedup(Max Size) = Ts/Tp
=1.9548/0.499697
= 3.911971

Efficiency = Speedup / p
= 3.911971 / 4
= 0.977992664

Processor: 4-core processor

Code:

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

int main(){
    double first, last;
    int t;
    int x;
    int i,j,k;
    int count = 1;
    printf("Enter number of tests : ");
    scanf("%d",&t);
    double sdifference[t],pdifference[t];
    for(x=0;x<t;x++)
    {
        int n;
        printf("Enter number of rows and columns for matrix %d : ",x+1);
        scanf("%d",&n);
        int a[n][n],b[n][n];
        int c[n][n];

    //accepting random values for matrices a and b
```

```
for(i = 0; i < n; i++)
        {
                for(j = 0; j < n; j++)
                {
                        a[i][j] = count;
                        count++;
                }
        }

        for(i = 0; i < n; i++)
        {
                for(j = 0; j < n; j++)
                {
                        b[i][j] = count;
                        count++;
                }
        }
   //first - time at which accepting values completes
   //last - time at which C(result matrix) is calculated
   //difference - difference b/w first and last
      first = omp_get_wtime(); //omp_get_wtime() gives the time

  // Multiplication of matrices
      for(i=0;i<n;i++)
      {
         for(j=0;j<n;j++)
         {
            c[i][j]=0;

            for(k=0;k<n;k++)
            {
               c[i][j]+=a[i][k]*b[k][j];
            }
         }
      }
      last = omp_get_wtime();
      sdifference[x] = last - first;

      first = omp_get_wtime();
      #pragma omp parallel for private(i,j,k)
      for(i=0;i<n;i++)
      {
         for(j=0;j<n;j++)
         {
            c[i][j]=0;
            for(k=0;k<n;k++)
            {
```

```
                c[i][j]+=a[i][k]*b[k][j];
            }
        }
    }
    last = omp_get_wtime();
    pdifference[x] = last - first;
  }

  printf("\n\nTEST CASE\t\tSERIAL DIFFERENCE\t\tPARALLEL DIFFERENCE\n");
  for(x=0;x<t;x++){
    printf("%d\t\t\t%lf\t\t\t%lf\n",x+1,sdifference[x],pdifference[x]);
  }
  return 0;
}
```

Output:

```
input
Enter number of tests : 5
Enter number of rows and columns for matrix 1 : 100
Enter number of rows and columns for matrix 2 : 200
Enter number of rows and columns for matrix 3 : 300
Enter number of rows and columns for matrix 4 : 400
5Enter number of rows and columns for matrix 5 :500


TEST CASE               SERIAL DIFFERENCE               PARALLEL DIFFERENCE
1                       0.007410                        0.020060
2                       0.030402                        0.007475
3                       0.108418                        0.157741
4                       0.517450                        0.532679
5                       0.685487                        1.000472


...Program finished with exit code 0
Press ENTER to exit console.
```

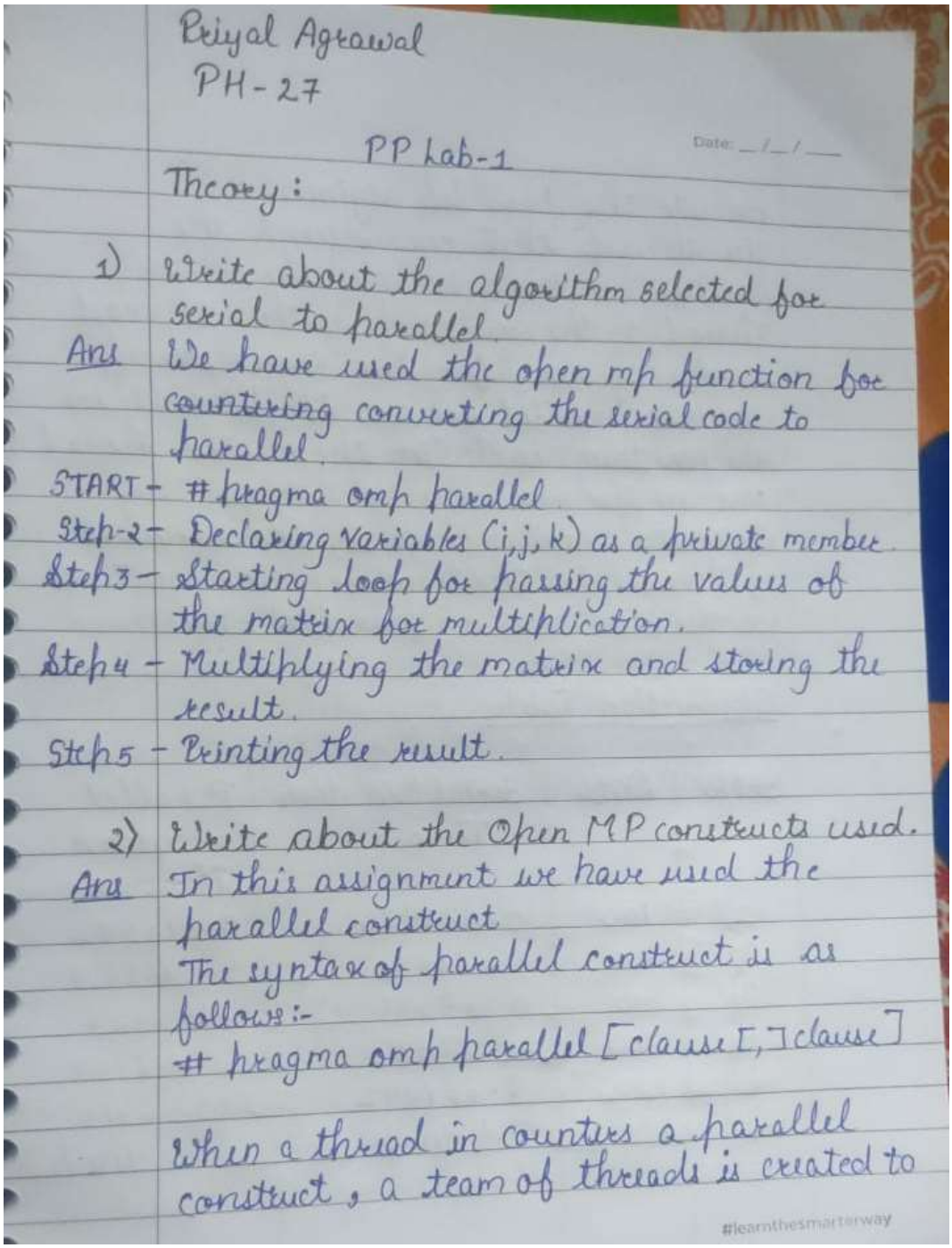
Input: Unsorted array of data points/values.

Output: Sorted Array of data points/values.

Platform: Online C compiler GDB

Conclusion: Thus, successfully studied, analyzed serial to parallel conversion.

FAQs:



Priyal Agrawal
PH-27

PP Lab-1

Date: __/__/____

Theory:

1) Write about the algorithm selected for serial to parallel.

Ans We have used the open mp function for converting the serial code to parallel.

START - #pragma omp parallel
Step-2 - Declaring variables (i,j,k) as a private member.
Step3 - Starting loop for passing the values of the matrix for multiplication.
Step4 - Multiplying the matrix and storing the result.
Step5 - Printing the result.

2) Write about the Open MP constructs used.

Ans In this assignment we have used the parallel construct
The syntax of parallel construct is as follows:-
#pragma omp parallel [clause[,]clause]

When a thread in counters a parallel construct, a team of threads is created to

#learnthesmarterway

Date: __/__/____

execute the parallel region.
The thread that encountered the parallel construct becomes the master thread of the new team, with a thread number of zero for the duration of the new parallel region. All threads in the new team, including the master thread execute the region.
Once the team is created, the number of threads in the team remains constant for the duration of parallel region.

Observation Table

Sr No	Data	Serial Execution Time	Parallel Execution Time.
1)	100	0.003780	0.010592
2)	200	0.065050	0.045824
3)	300	0.105033	0.194089
4)	400	0.262550	0.446177
5)	500	2.661442	1.106417
6)	800	Code dump	Code dump

Date: __/__/____

FAQ's -

1) What is an Open MP directive? Give syntax of parallel for directive.

Ans Open MP is a set of compiler directives as well as an API for Programs written in C, C++ or FORTAN that provides support for parallel programming in shared memory. Open MP identifies parallel region blocks of code that may run in parallel. Application developers insert compiler directions into their code at parallel region, and these directive instruct the open-MP run-time library to execute the parallel in the region.

Syntax of parallel for directive

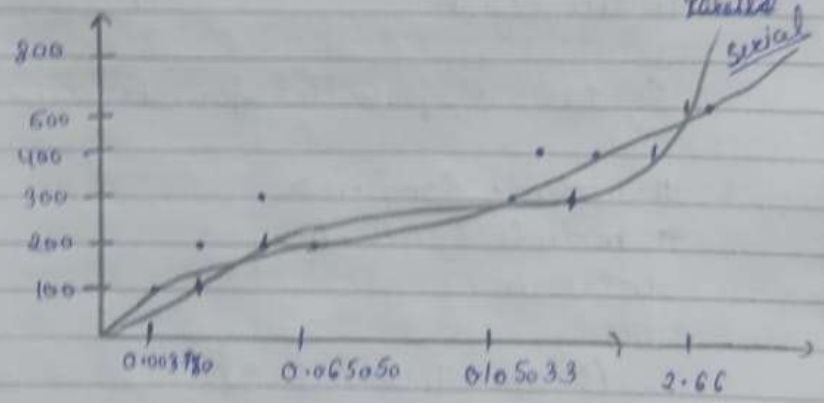
```
# include <omp.h>
# include <stdio.h>
 int main ()
 {
   /* Sequential Code */
```

#learnthesmartsway

```
#pragma omp parallel { ... }
  return 0;
}
```

2) Give an example from your daily activities where you can incorporate parallelism.

Ans
- Mother was very busy gathering the laundry, dusting the furniture, & washing the dishes.
- He likes television shows that the have deep characters, interesting stories, and good actors.
- My face is washed, my hair is combed, and my teeth are brushed.



Name: Vatsal Board
Roll No: 23
Batch: P12

PP Assignment: 1

FAQs

1) What is an OpenMP directive? Give Syntax of parallel for directive.

=> OpenMP is a set of compiler directives as well as an API for programs that may be used to explicity direct multi threaded, shared memory parallelism.

=> Parallel directive defines a parallel region, which is code that will be executed by multiple threads in parallel.

Syntax:

```
#pragma omp parallel [clauses]
{
    // code block
}
```

2) Give an example from your daily activity where you can incorporate parallelism.

-> Watching movie with parallel activity like seeing and listening.
-> In Gym doing exercise and Listening Music.
-> In Lecture, Understanding and taking Notes.