Text

Description automatically generated

**T.Y.B.Tech (CSE)**

High Performance Computing

**Lab Assignment No – 2**

**Name: Aniruddha Shende**

**Roll number: PE04**

**Batch: E1**

**Panel: E**

**Assignment no: 2**

**Title: Serial to Parallel**

**Aim: To write a parallel program for an existing serial program**

**Objective:**

1. **To analyze the existing algorithm**
2. **To use appropriate OpenMP directive to parallelize the code.**

**Theory:**

1. **Serial program and Parallelization of the program.**

**Serial program is executed in a sequential manner whereas using parallel programming, your code spreads data processing tasks across multiple CPUs for radically better performance.**

**The main difference between serial and parallel processing in computer architecture is that serial processing performs a single task at a time while parallel processing performs is multiple tasks at a time. Therefore, the performance of parallel processing is higher than in serial processing.**

**FAQ:  
 1. What are OpenMP directives?**

**Ans 1: OpenMP directives exploit shared memory parallelism by defining various types of parallel regions.**

**OpenMP provides compiler directives and clauses that used together control the parallelism of regions of code. The directive keyword specifies the parallel action you want to take place.**

**The format for OpenMP compiler directives and clauses is as follows:**

**#pragma omp directive [clause[[.] clause]...] newline**

**We can use the various OpenMP directives like parallel, for, sections, single, parallel for, parallel, sections, barrier, flush, master, critical, atomic, ordered, etc.**

1. **Give an example from your daily activities where you can incorporate parallelism.**

**Ans 2: Some of the daily activites in which we incorporate parallelism Eating dinner while watching TV and cooking food while cleaning house.**

**Conclusion:**

**Successfully converted a serial program to parallel program. After performing the given experiment we can see that after making the program, a parallel program using the header “omp.h” we see that the execution time of the program becomes less as compared to the execution time of the same serial program.**

**Table :**

|  |  |  |
| --- | --- | --- |
| **Data points** | **Serial execution time** | **Parallel execution time** |
| **2500** | **0.983734** | **6.009051** |
| **10000** | **8.849930** | **9.926322** |
| **22500** | **22.475784** | **12.926322** |
| **40000** | **34.950494** | **17.677064** |
| **62500** | **51.117142** | **24.261547** |
| **90000** | **84.669420** | **28.155339** |
| **122500** | **125.778970** | **56.816012** |

**Plot :**

**Chart, line chart

Description automatically generated**

**Program Code :**

// Name : Aniruddha Shende

// Roll no : PE04

// Batch : E1

// Panel : E

#include <stdio.h>

#include <pthread.h>

#include <stdlib.h>

#include <sys/time.h>

#include <omp.h>

int main()

{

int r1, c1, r2, c2, i, j, k;

printf("Enter 1st matrix rows : ");

scanf("%d", &r1);

printf("Enter 1st matrix columns : ");

scanf("%d", &c1);

int a[r1][c1];

//printf("Enter the elements of First matrix\n");

for (i = 0; i < r1; i++)

{

for (j = 0; j < c1; j++)

{

//assigning a random value

a[i][j] = 3000;

// scanf("%d", &a[i][j]);

}

}

printf("\n");

printf("Enter 2nd matrix rows : ");

scanf("%d", &r2);

printf("Enter 2nd matrix columns : ");

scanf("%d", &c2);

int b[r2][c2];

int final\_multiplication[r1][c2];

printf("Enter the elements of Second matrix\n");

for (i = 0; i < r2; i++)

{

for (j = 0; j < c2; j++)

{­

a /z/assigning a random value

b[i][j] = 3000;

// scanf("%d", &a[i][j]);

}

}

printf("\n");

if (c1 != r2 || r1 < 0 || c1 < 0 || r2 < 0 || c2 < 0)

{

printf("\nMatrix multiplication not possible\n");

}

// for serial

double start = omp\_get\_wtime();

#pragma omp parallel for

for (i = 0; i < r1; i++)

{­­­

for (j = 0; j < c2; j++)

{

final\_multiplication[i][j] = 0;

for (k = 0; k < r2; k++)

{

final\_multiplication[i][j] += a[i][k] \* b[k][j];

}

// printf("%d ", final\_multiplication[i][j]);

}

// printf("\n");

}

double end = omp\_get\_wtime();

printf("\nTime taken for Serial: %lf\n", (end-start)\*1000);

//for parallel

start = omp\_get\_wtime();

#pragma omp parallel for

for (i = 0; i < r1; i++)

{

for (j = 0; j < c2; j++)

{

final\_multiplication[i][j] = 0;

for (k = 0; k < r2; k++)

{

final\_multiplication[i][j] += a[i][k] \* b[k][j];

}

// printf("%d ", final\_multiplication[i][j]);

}

// printf("\n");

}

end = omp\_get\_wtime();

printf("\nTime taken for Parallel: %lf\n", (end-start)\*1000);

// for (i = 0; i < r1; i++)

// {

// for (j = 0; j < c2; j++)

// {

// printf("%d\t", final\_multiplication[i][j]);

// }

// printf("\n");

// }

// return 0;

}

**Execution Screenshot :**

Text

Description automatically generated