Walchand College of Engineering, Sangli
Department of Computer Science and Engineering

Class: Final Year (Computer Science and Engineering)

Year: 2023-24 **Semester:** 1

Course: High Performance Computing Lab

Practical No. 2

Exam Seat No: 2020BTECS00037

Title of practical: Study and implementation of basic OpenMP clauses

Implement following Programs using OpenMP with C:

1. Vector Scalar Addition
2. Calculation of value of Pi
   Analyse the performance of your programs for different number of threads and Data size.

Problem Statement 1:

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

#define VECTOR_SIZE 10000

int main() {
    float vector[VECTOR_SIZE];
    int scalar = 5;
    for (int i = 0; i < VECTOR_SIZE; i++) {
        vector[i] = i + 100.987;
    }

    double start_time_serial = omp_get_wtime();
    for (int i = 0; i < VECTOR_SIZE; i++) {
        vector[i] += scalar;
    }

    double end_time_serial = omp_get_wtime();
    printf("Serial Method Time: %f seconds\n", (end_time_serial -
start_time_serial));

    for (int i = 0; i < VECTOR_SIZE; i++) {
        vector[i] = i + 100.987;
    }
```

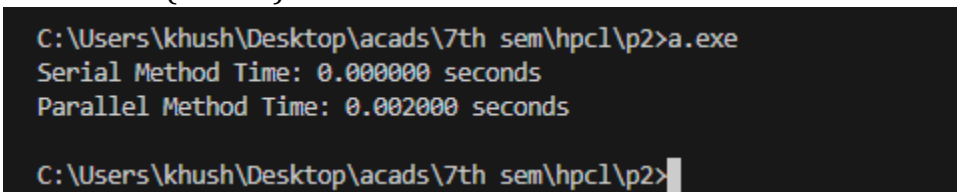
```
    double start_time_parallel = omp_get_wtime();

    #pragma omp parallel for private(scalar) num_threads(100)
        for (int i = 0; i < VECTOR_SIZE; i++) {
                vector[i] += scalar;
            }
    double end_time_parallel = omp_get_wtime();
    printf("Parallel Method Time: %f seconds\n", (end_time_parallel -
start_time_parallel));
    return 0;
}
```

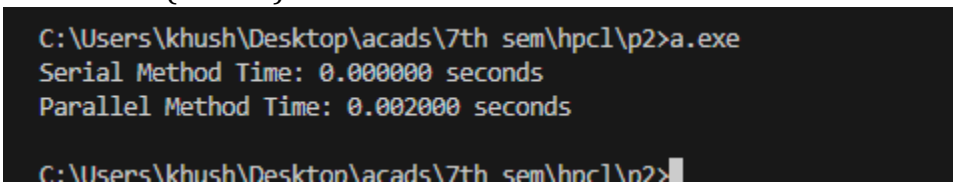
Screenshots:

Keeping number of threads constant and varying size of Data.

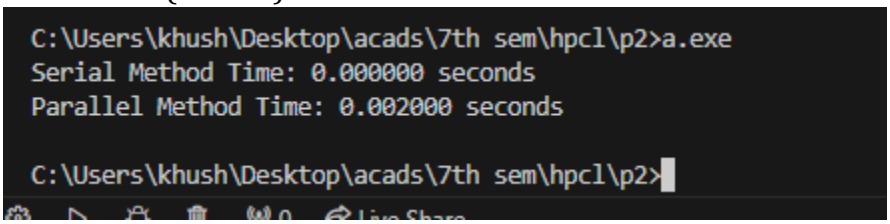
Threads = 8(default) Vector size = 100



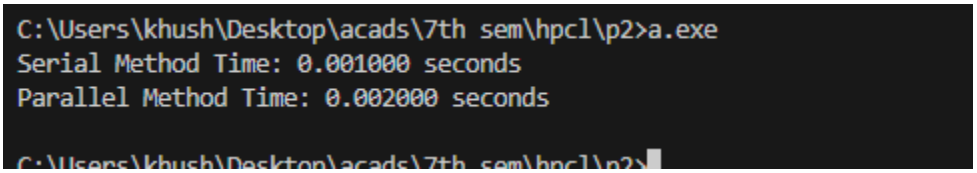
Threads = 8(default) Vector size = 1000



Threads = 8(default) Vector size = 10000

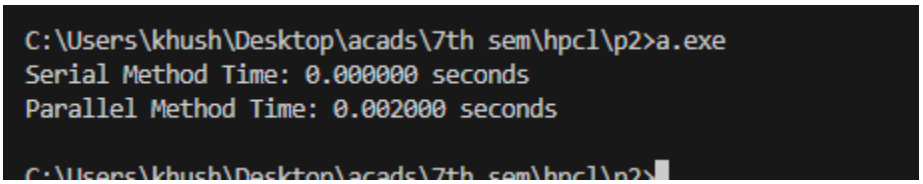


Threads = 8(default) Vector size = 100000

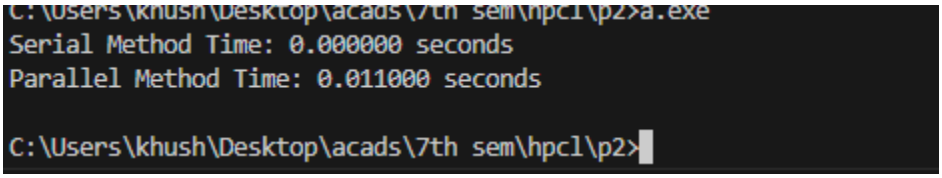


Keeping data constant and increasing number of threads.

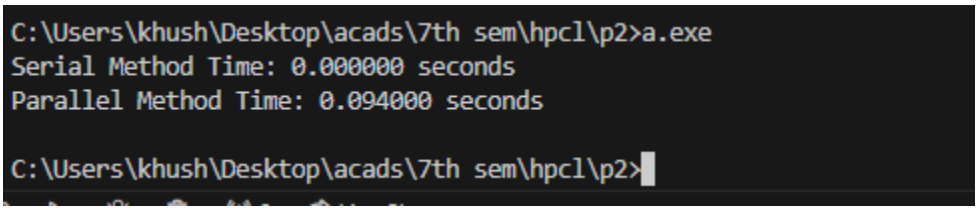
Threads = 10 Vector size = 10000



Threads = 100 Vector size = 10000



Threads = 1000 Vector size = 10000



Information:

Vector and scaler addition is to be performed using sequential and parallel approach. We have to analyse the time both approaches. For parallel approach analysis can be done in two ways, first by keeping data constant and varying number of threads and secondly by keeping number of threads constant and varying size of data.

Analysis:

1) As we go on increasing the size of data the time it takes to execute in parallel also increases.
2) By keeping data constant and increasing number for threads gradually increase the execution time due to increase in logical thread causes extra mapping time.
3) Here Serial time is less than parallel because insufficient data for parallelism which causes extra overhead of communication time.

Number of Threads	Data Size	Sequential Time	Parallel Time
8	100	0.00000	0.002000
8	1000	0.00000	0.002000
8	10000	0.00000	0.002000
8	100000	0.00000	0.002000
10	10000	0.00000	0.002000
100	10000	0.00000	0.011000
1000	10000	0.00000	0.094000

Problem Statement 2:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>
int main() {
    int totalPoints = 10000000;
    int pointsInsideCircle = 0;
    double x, y;
    printf("Enter the number of terms: ");
    scanf("%d", &totalPoints);
    double start_time_serial = omp_get_wtime();
    for (int i = 0; i < totalPoints; ++i) {
        x = (double)rand() / RAND_MAX;
        y = (double)rand() / RAND_MAX;
        if (x * x + y * y <= 1.0) {
            pointsInsideCircle++;
        }
    }
    double pi = 4.0 * pointsInsideCircle / totalPoints;
    double end_time_serial = omp_get_wtime();
    printf("serial Method Time: %f seconds\n", (end_time_serial -
start_time_serial));
    printf("Estimated value of pi: %f\n", pi);
    return 0;
}
```

Screenshots:

```
C:\Users\khush\Desktop\acads\7th sem\hpcl\p2>a.exe
Enter the number of terms: 8000
serial Method Time: 0.000000 seconds
Estimated value of pi: 3.110000
```

```
C:\Users\khush\Desktop\acads\7th sem\hpcl\p2>a.exe
Enter the number of terms: 980000
serial Method Time: 0.027000 seconds
Estimated value of pi: 3.139412

C:\Users\khush\Desktop\acads\7th sem\hpcl\p2>
```

```
C:\Users\khush\Desktop\acads\7th sem\hpcl\p2>a.exe
Enter the number of terms: 7840005
serial Method Time: 0.271000 seconds
Estimated value of pi: 3.141408
```

```
Enter the number of terms: 500000
serial Method Time: 0.015000 seconds
Estimated value of pi: 3.141160
```

Parallel

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>
int main() {
    int totalPoints = 10000000;
    int pointsInsideCircle = 0;
    double x, y;
    printf("Enter the number of terms: ");
    scanf("%d", &totalPoints);
    double start_time_parallel = omp_get_wtime();
    #pragma omp parallel for private(x, y) reduction(+:pointsInsideCircle)
    for (int i = 0; i < totalPoints; ++i) {
        x = (double)rand() / RAND_MAX;
        y = (double)rand() / RAND_MAX;
        if (x * x + y * y <= 1.0) {
            pointsInsideCircle++;
        }
    }
    double pi = 4.0 * pointsInsideCircle / totalPoints;
    double end_time_parallel = omp_get_wtime();
    printf("Parallel Method Time: %f seconds\n", (end_time_parallel -
start_time_parallel));
    printf("Estimated value of pi: %f\n", pi);
    return 0;
}
```

OUTPUT

```
C:\Users\khush\Desktop\acads\7th sem\hpcl\p2>a.exe
Serial Method Time: 0.000000 seconds
Parallel Method Time: 0.094000 seconds

C:\Users\khush\Desktop\acads\7th sem\hpcl\p2>gcc -fopenmp a2q2.c

C:\Users\khush\Desktop\acads\7th sem\hpcl\p2>a.exe
Enter the number of terms: 90000
serial Method Time: 0.002000 seconds
Estimated value of pi: 3.136089

C:\Users\khush\Desktop\acads\7th sem\hpcl\p2>gcc -fopenmp a2q2.c

C:\Users\khush\Desktop\acads\7th sem\hpcl\p2>a.exe
Enter the number of terms: 80000
serial Method Time: 0.002000 seconds
Estimated value of pi: 3.135200

C:\Users\khush\Desktop\acads\7th sem\hpcl\p2>gcc -fopenmp a2q2.c

C:\Users\khush\Desktop\acads\7th sem\hpcl\p2>a.exe
Enter the number of terms: 3000
serial Method Time: 0.000000 seconds
Estimated value of pi: 3.092000
```