Practical 3

PRN: 23520005
Name: Manas Indrapal Gedam
Batch: B7

GitHub Link: Link

Study and Implementation of schedule, nowait, reduction, ordered and collapse clauses

Problem Statement 1:

Analyse and implement a Parallel code for below program using OpenMP.

// C Program to find the minimum scalar product of two vectors (dot product)

To get the **minimum scalar (dot) product**, we:

1. **Sort one vector in ascending order**
2. **Sort the other vector in descending order**
3. Multiply corresponding elements and sum the result

Program:

C P1.c > ...

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

// Comparison functions for qsort
int compare_asc(const void *a, const void *b) {
    return (*(int*)a - *(int*)b);
}
int compare_desc(const void *a, const void *b) {
    return (*(int*)b - *(int*)a);
}

int main() {
    int n = 1000000;
    int *A = malloc(n * sizeof(int));
    int *B = malloc(n * sizeof(int));
    long long result = 0;

    // Initialize vectors with some values
    for (int i = 0; i < n; i++) {
        A[i] = rand() % 100;
        B[i] = rand() % 100;
    }

    // Sort A ascending, B descending
    qsort(A, n, sizeof(int), compare_asc);
    qsort(B, n, sizeof(int), compare_desc);

    double start = omp_get_wtime();

    // Parallel dot product using reduction
    #pragma omp parallel for reduction(+:result) num_threads(4)
    for (int i = 0; i < n; i++) {
        result += (long long)A[i] * B[i];
    }

    double end = omp_get_wtime();
```

```
    printf("(char [24])"Time Taken: %f seconds\n");
    printf("Time Taken: %f seconds\n", end - start);

    free(A);
    free(B);
    return 0;
}
```

Output:

```
manas@Manass-MacBook-Air HPCL 3 % gcc-15 -fopenmp P1.c -o P1
manas@Manass-MacBook-Air HPCL 3 % ./P1
Minimum Scalar Product: 1615929279
Time Taken: 0.000711 seconds
manas@Manass-MacBook-Air HPCL 3 %
```

Information

- Used clauses: parallel for, reduction, num_threads
- Sort is serial (qsort is fast); dot product is parallelized

Analysis:

Parallel reduction improves performance as vector size increases

Near-linear speedup until memory/cache saturation

Sorting is a one-time step; dot product is bottleneck for large n