Day8

Table of Contents

1. matrix addition

```
#include<stdio.h>
#include<omp.h>
#include<stdlib.h>
#define N 10000
#define T 13

int main(){
    int **a, **b, **c;
    a = (int**) malloc(sizeof(int*) * N);
    b = (int**) malloc(sizeof(int*) * N);
    c = (int**) malloc(sizeof(int*) * N);

    for(int i = 0; i < N; i++){
        a[i] = (int*) malloc(sizeof(int) * N);
        b[i] = (int*) malloc(sizeof(int) * N);
        c[i] = (int*) malloc(sizeof(int) * N);
    }

    for(int i = 0; i < N; i++){
        for(int j = 0; j < N; j++){
            a[i][j] = i + 1;
```

```
            b[i][j] = i + 1;
            c[i][j] = 0;
        }
    }

    double startTime = omp_get_wtime();
    #pragma omp parallel for num_threads(T)
    for(int i = 0; i < N; i++){
        for(int j = 0; j < N; j++){
            c[i][j] = a[i][j] + b[i][j];
        }
    }
    double endTime = omp_get_wtime();
    double parallelTime = endTime - startTime;

    /*
    for(int i = N - 1; i < N; i++){
        for(int j = N - 50; j < N; j++){
            printf("%d\t", c[i][j]);
        }
        printf("\n");
    }
    */

    printf("%lf\n", parallelTime);

    for(int i = 0; i < N; i++){
        free(a[i]);
        free(b[i]);
        free(c[i]);
    }
    free(a);
    free(b);
    free(c);

    return 0;
}
```

```
gcc matrixAddition.c -fopenmp -o matrixAddition.out
```

```
#./matrixAddition.out > output1.txt
#echo "check output1.txt"
./matrixAddition.out
```

```
0.059650
```

2. scheduling

- static
- dynamic
- guided
- auto
- runtime

3. static

```
#pragma omp parallel for schedule(static, chunksize)
```

```
#include<stdio.h>
#include<omp.h>
#define N 20
#define T 6
int main(){
    #pragma omp parallel for schedule(static, 3) num_threads(T)
    for(int i = 0; i < N; i++){
        printf("thread\t%d\t:\ti\t%d\n", omp_get_thread_num(), i);
    }
}
```

```
gcc static.c -fopenmp -o static.out
```

```
./static.out
```

```
thread  5       :       i       15
thread  5       :       i       16
thread  5       :       i       17
thread  4       :       i       12
```

```
thread  4       :       i       13
thread  3       :       i       9
thread  3       :       i       10
thread  3       :       i       11
thread  4       :       i       14
thread  2       :       i       6
thread  0       :       i       0
thread  0       :       i       1
thread  0       :       i       2
thread  0       :       i       18
thread  0       :       i       19
thread  2       :       i       7
thread  2       :       i       8
thread  1       :       i       3
thread  1       :       i       4
thread  1       :       i       5
```

4. dynamic

```
#pragma omp parallel for schedule(dynamic, chunksize)
```

```
#include<stdio.h>
#include<omp.h>
#define N 20
#define T 6
int main(){
    #pragma omp parallel for schedule(dynamic, 3) num_threads(T)
    for(int i = 0; i < N; i++){
        printf("thread\t%d\t:\ti\t%d\n", omp_get_thread_num(), i);
    }
}
```

```
gcc dynamic.c -fopenmp -o dynamic.out
```

```
./dynamic.out
```

```
thread  2       :       i       0
thread  2       :       i       1
```

```
thread  2       :       i       2
thread  2       :       i       9
thread  2       :       i       10
thread  2       :       i       11
thread  2       :       i       12
thread  2       :       i       13
thread  2       :       i       14
thread  2       :       i       15
thread  2       :       i       16
thread  2       :       i       17
thread  2       :       i       18
thread  2       :       i       19
thread  1       :       i       3
thread  1       :       i       4
thread  1       :       i       5
thread  4       :       i       6
thread  4       :       i       7
thread  4       :       i       8
```

5. dynamic2

```c
#include<stdio.h>
#include<omp.h>
#define N 10
#define T 5
int main(){
    int a[N] = {1343, 100, 500000, 322, 4444, 544, 300, 70000000, 400, 3244};
    #pragma omp parallel for schedule(dynamic, 1) num_threads(T)
    for(int i = 0; i < N; i++){
        printf("iteration i = %d is assigned to %d\n", i, omp_get_thread_num());
        for(int j = 0; j < a[i]; j++);
    }
}
```

```
gcc dynamic2.c -fopenmp -o dynamic2.out
```

```
./dynamic2.out
```

```
iteration i = 0 is assigned to 4
```

```
iteration i = 4 is assigned to 2
iteration i = 5 is assigned to 4
iteration i = 6 is assigned to 4
iteration i = 1 is assigned to 1
iteration i = 8 is assigned to 2
iteration i = 9 is assigned to 2
iteration i = 2 is assigned to 0
iteration i = 3 is assigned to 3
iteration i = 7 is assigned to 4
```

6. guided

```
#pragma omp parallel for schedule(guided, chunksize)
```

```
#include<stdio.h>
#include<omp.h>
#define N 20
#define T 3
int main(){
    #pragma omp parallel for schedule(guided, 3) num_threads(T)
    for(int i = 0; i < N; i++){
        printf("thread\t%d\t:\ti\t%d\n", omp_get_thread_num(), i);
    }
}
```

```
gcc guided.c -fopenmp -o guided.out
```

```
./guided.out
```

```
thread  1       :       i       0
thread  1       :       i       1
thread  1       :       i       2
thread  1       :       i       3
thread  1       :       i       4
thread  1       :       i       5
thread  1       :       i       6
thread  1       :       i       15
thread  2       :       i       12
```

```
thread  0       :       i       7
thread  0       :       i       8
thread  0       :       i       9
thread  0       :       i       10
thread  0       :       i       11
thread  0       :       i       18
thread  0       :       i       19
thread  2       :       i       13
thread  2       :       i       14
thread  1       :       i       16
thread  1       :       i       17
```

7. guided2

```c
#include<stdio.h>
#include<omp.h>
#define N 10
#define T 5
int main(){
    int a[N] = {1343, 100, 500000, 322, 4444, 544, 300, 70000000, 400, 3244};
    #pragma omp parallel for schedule(dynamic, 1) num_threads(T)
    for(int i = 0; i < N; i++){
        printf("iteration i = %d is assigned to %d\n", i, omp_get_thread_num());
        for(int j = 0; j < a[i]; j++);
    }
}
```

```
gcc guided2.c -fopenmp -o guided2.out
```

```
./guided2.out
```

```
iteration i = 0 is assigned to 3
iteration i = 3 is assigned to 3
iteration i = 4 is assigned to 3
iteration i = 1 is assigned to 1
iteration i = 5 is assigned to 1
iteration i = 6 is assigned to 1
iteration i = 7 is assigned to 1
iteration i = 8 is assigned to 3
```

```
iteration i = 9 is assigned to 3
iteration i = 2 is assigned to 0
```

8. runtime

```
#pragma omp parallel for schedule(runtime)
```

```
#include<stdio.h>
#include<omp.h>
#define N 20
#define T 6
int main(){
    #pragma omp parallel for schedule(runtime) num_threads(T)
    for(int i = 0; i < N; i++){
        printf("thread\t%d\t:\ti\t%d\n", omp_get_thread_num(), i);
    }
}
```

```
gcc runtime.c -fopenmp -o runtime.out
```

```
export OMP_SCHEDULE="dynamic,3"
./runtime.out
```

```
thread  5       :       i       0
thread  5       :       i       1
thread  5       :       i       2
thread  5       :       i       3
thread  5       :       i       4
thread  5       :       i       5
thread  5       :       i       6
thread  5       :       i       7
thread  5       :       i       8
thread  5       :       i       9
thread  5       :       i       10
thread  5       :       i       11
thread  5       :       i       12
thread  5       :       i       13
thread  5       :       i       14
```

```
thread  5       :       i       15
thread  5       :       i       16
thread  5       :       i       17
thread  5       :       i       18
thread  5       :       i       19
```

9. omp_set_schedule

```
// To use this function we need to use schedule(runtime) clause
type : omp_sched_static, omp_sched_dynamic, omp_sched_guided
omp_set_schedule(type, chunksize);
```

```
#include<stdio.h>
#include<omp.h>
#define N 20
#define T 6
int main(){
    omp_set_schedule(omp_sched_dynamic, 8);
    #pragma omp parallel for schedule(runtime) num_threads(T)
    for(int i = 0; i < N; i++){
        printf("thread\t%d\t:\ti\t%d\n", omp_get_thread_num(), i);
    }
}
```

```
gcc sched.c -fopenmp -o sched.out
```

```
./sched.out
```

```
thread  5       :       i       0
thread  5       :       i       1
thread  5       :       i       2
thread  5       :       i       3
thread  5       :       i       4
thread  5       :       i       5
thread  5       :       i       6
thread  5       :       i       7
thread  3       :       i       16
thread  3       :       i       17
```

```
thread  3       :       i       18
thread  3       :       i       19
thread  0       :       i       8
thread  0       :       i       9
thread  0       :       i       10
thread  0       :       i       11
thread  0       :       i       12
thread  0       :       i       13
thread  0       :       i       14
thread  0       :       i       15
```

Author: Abhishek Raj
Created: 2025-01-03 Fri 12:03