

UNIVERSITÉ IBN TOFAIL

FACULTÉ DES SCIENCES - KENITRA

DÉPARTEMENT INFORMATIQUE

MASTER INFORMATIQUE ET INTELLIGENCE ARTIFICIELLE

COMPTE RENDU

TP 3 : Génération de l'Ensemble de Mandelbrot avec OpenMP

Réalisé par :

Anas BOUKHLIJA

Encadré par :

Pr. Nada FAQIR

ANNÉE UNIVERSITAIRE 2024-2025

Table des matières

1	Code Séquentiel pour Générer l'Ensemble de Mandelbrot	3
2	Parallélisation avec OpenMP	5
3	Analyse des Performances	7
3.1	Temps d'exécution séquentiel	7
3.2	Temps d'exécution parallèle	7
3.3	Temps d'exécution parallèle avec différents nombres de threads	7
3.4	Analyse de l'Impact de la Gestion des Tâches sur la Performance	10
3.4.1	Performance avec Différents Nombres de Threads :	10
3.4.2	Gestion des Tâches :	10
4	Optimisation des Tâches	11

Table des figures

1	Temps d'exécution séquentiel	7
2	Temps d'exécution parallèle	7
3	Temps d'exécution parallèle avec différents nombres de threads	9
4	Temps d'exécution parallèle avec optimisation des Tâches	12

1 Code Séquentiel pour Générer l'Ensemble de Mandelbrot

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <complex.h>
4 #include <omp.h>
5
6 #define WIDTH 1600
7 #define HEIGHT 1600
8 #define MAX_ITER 1000
9
10 int mandelbrot(double complex c) {
11     double complex z = 0 + 0*I;
12     int iter;
13     for (iter = 0; iter < MAX_ITER; iter++) {
14         if (cabs(z) > 2.0) break;
15         z = z*z + c;
16     }
17     return iter;
18 }
19
20 void generate_mandelbrot(unsigned char image[HEIGHT][WIDTH]) {
21     for (int y = 0; y < HEIGHT; y++) {
22         for (int x = 0; x < WIDTH; x++) {
23             double complex c = (x - WIDTH/2.0)*4.0/WIDTH + (y -
HEIGHT/2.0)*4.0/HEIGHT*I;
24             int value = mandelbrot(c);
25             image[y][x] = (value == MAX_ITER) ? 0 : (255 * value /
MAX_ITER);
26         }
27     }
28 }
29
30 int main() {
31     double start, end;
32     unsigned char image[HEIGHT][WIDTH];
33 }
```

```
34     start = omp_get_wtime();
35
36     generate_mandelbrot(image);
37
38     end = omp_get_wtime();
39
40     printf("Temps d'execution : %f secondes\n", end - start);
41
42     FILE *fp = fopen("/home/anas/Desktop/mandelbrot.pgm", "wb");
43
44     fprintf(fp, "P5\n%d %d\n255\n", WIDTH, HEIGHT);
45     fwrite(image, 1, WIDTH*HEIGHT, fp);
46     fclose(fp);
47     printf("Image genereee : mandelbrot.pgm\n");
48     return 0;
49 }
```

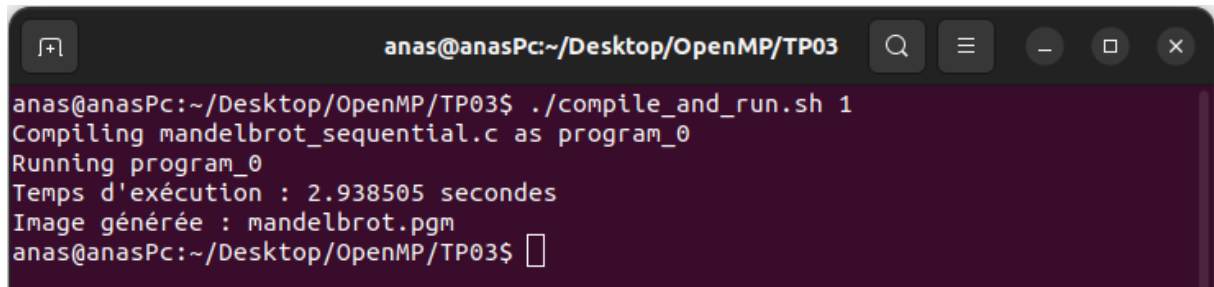
2 Parallélisation avec OpenMP

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <complex.h>
4 #include <omp.h>
5
6 #define WIDTH 1600
7 #define HEIGHT 1600
8 #define MAX_ITER 1000
9
10 int mandelbrot(double complex c) {
11     double complex z = 0 + 0*I;
12     int iter;
13     for (iter = 0; iter < MAX_ITER; iter++) {
14         if (cabs(z) > 2.0) break;
15         z = z*z + c;
16     }
17     return iter;
18 }
19
20 void generate_mandelbrot(unsigned char image[HEIGHT][WIDTH]) {
21     #pragma omp parallel
22     {
23         #pragma omp single
24         for (int y = 0; y < HEIGHT; y++) {
25             for (int x = 0; x < WIDTH; x++) {
26                 #pragma omp task
27                 {
28                     double complex c = (x - WIDTH/2.0)*4.0/WIDTH +
29                     (y - HEIGHT/2.0)*4.0/HEIGHT*I;
30                     int value = mandelbrot(c);
31                     image[y][x] = (value == MAX_ITER) ? 0 : (255 *
32                     value / MAX_ITER);
33                 }
34             }
35         }
36     }
```

```
34     }
35 }
36
37 int main() {
38     double start, end;
39     unsigned char image[HEIGHT][WIDTH];
40
41     start = omp_get_wtime();
42
43     generate_mandelbrot(image);
44
45     end = omp_get_wtime();
46
47     printf("Temps d'execution : %f secondes\n", end - start);
48
49     FILE *fp = fopen("/home/anas/Desktop/mandelbrot.pgm", "wb");
50
51     fprintf(fp, "P5\n%d %d\n255\n", WIDTH, HEIGHT);
52     fwrite(image, 1, WIDTH*HEIGHT, fp);
53     fclose(fp);
54     printf("Image generee : mandelbrot.pgm\n");
55     return 0;
56 }
```

3 Analyse des Performances

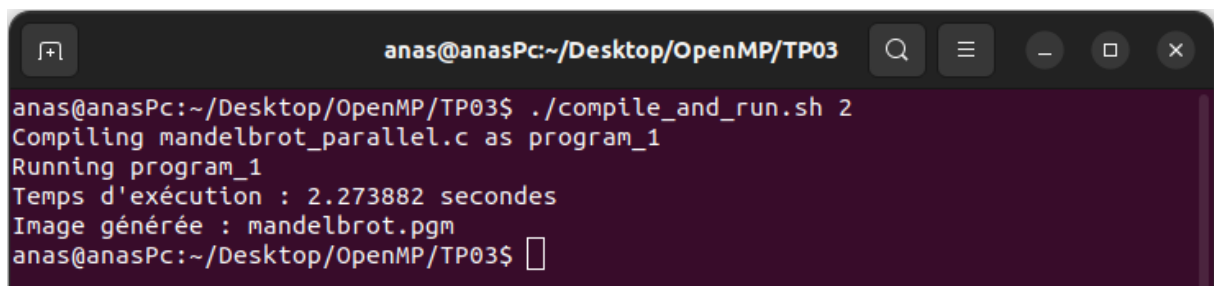
3.1 Temps d'exécution séquentiel

A terminal window titled 'anas@anasPc:~/Desktop/OpenMP/TP03' showing the execution of a sequential Mandelbrot program. The user runs './compile_and_run.sh 1', which compiles 'mandelbrot_sequential.c' as 'program_0' and runs it. The output shows an execution time of 2.938505 seconds and the generation of 'mandelbrot.pgm'.

```
anas@anasPc:~/Desktop/OpenMP/TP03$ ./compile_and_run.sh 1
Compiling mandelbrot_sequential.c as program_0
Running program_0
Temps d'exécution : 2.938505 secondes
Image générée : mandelbrot.pgm
anas@anasPc:~/Desktop/OpenMP/TP03$
```

FIGURE 1 – Temps d'exécution séquentiel

3.2 Temps d'exécution parallèle

A terminal window titled 'anas@anasPc:~/Desktop/OpenMP/TP03' showing the execution of a parallel Mandelbrot program. The user runs './compile_and_run.sh 2', which compiles 'mandelbrot_parallel.c' as 'program_1' and runs it. The output shows an execution time of 2.273882 seconds and the generation of 'mandelbrot.pgm'.

```
anas@anasPc:~/Desktop/OpenMP/TP03$ ./compile_and_run.sh 2
Compiling mandelbrot_parallel.c as program_1
Running program_1
Temps d'exécution : 2.273882 secondes
Image générée : mandelbrot.pgm
anas@anasPc:~/Desktop/OpenMP/TP03$
```

FIGURE 2 – Temps d'exécution parallèle

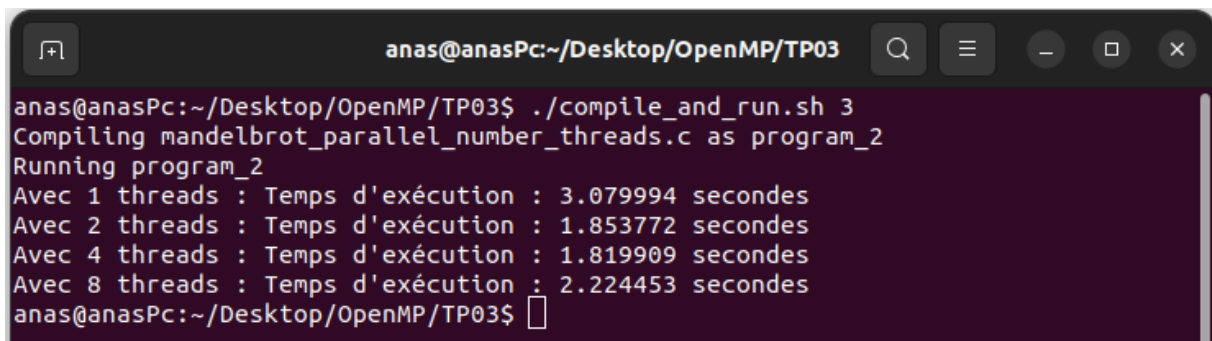
3.3 Temps d'exécution parallèle avec différents nombres de threads

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <complex.h>
4 #include <omp.h>
5
6 #define WIDTH 1600
7 #define HEIGHT 1600
8 #define MAX_ITER 1000
9
10 int mandelbrot(double complex c) {
```



```
11     double complex z = 0 + 0*I;
12     int iter;
13     for (iter = 0; iter < MAX_ITER; iter++) {
14         if (cabs(z) > 2.0) break;
15         z = z*z + c;
16     }
17     return iter;
18 }
19
20 void generate_mandelbrot(unsigned char image[HEIGHT][WIDTH]) {
21     #pragma omp parallel
22     {
23         #pragma omp single
24         for (int y = 0; y < HEIGHT; y++) {
25             for (int x = 0; x < WIDTH; x++) {
26                 #pragma omp task
27                 {
28                     double complex c = (x - WIDTH/2.0)*4.0/WIDTH +
29                     (y - HEIGHT/2.0)*4.0/HEIGHT*I;
30                     int value = mandelbrot(c);
31                     image[y][x] = (value == MAX_ITER) ? 0 : (255 *
32                     value / MAX_ITER);
33                 }
34             }
35         }
36     }
37
38 int main() {
39     unsigned char image[HEIGHT][WIDTH];
40
41     // Test with different numbers of threads
42     for (int num_threads = 1; num_threads <= 8; num_threads*=2) {
43         omp_set_num_threads(num_threads);
44
45         double start = omp_get_wtime();
```

```
46     generate_mandelbrot(image);
47
48     double end = omp_get_wtime();
49
50     FILE *fp = fopen("/home/anas/Desktop/mandelbrot.pgm", "wb"
51 );
52     fprintf(fp, "P5\n%d %d\n255\n", WIDTH, HEIGHT);
53     fwrite(image, 1, WIDTH * HEIGHT, fp);
54     fclose(fp);
55
56     // Print the execution time for each thread count
57     printf("Avec %d threads : Temps d'exécution : %f secondes\n", num_threads, end - start);
58 }
59
60 return 0;
61 }
```



A terminal window titled 'anas@anasPc:~/Desktop/OpenMP/TP03' showing the output of a script. The script compiles and runs a program for 1, 2, 4, and 8 threads. The execution times are displayed in French. The times for 1, 2, and 4 threads are approximately 3.08, 1.85, and 1.82 seconds respectively, while for 8 threads it is 2.22 seconds.

```
anas@anasPc:~/Desktop/OpenMP/TP03$ ./compile_and_run.sh 3
Compiling mandelbrot_parallel_number_threads.c as program_2
Running program_2
Avec 1 threads : Temps d'exécution : 3.079994 secondes
Avec 2 threads : Temps d'exécution : 1.853772 secondes
Avec 4 threads : Temps d'exécution : 1.819909 secondes
Avec 8 threads : Temps d'exécution : 2.224453 secondes
anas@anasPc:~/Desktop/OpenMP/TP03$
```

FIGURE 3 – Temps d'exécution parallèle avec différents nombres de threads

3.4 Analyse de l'Impact de la Gestion des Tâches sur la Performance

3.4.1 Performance avec Différents Nombres de Threads :

- **Un seul thread :** Temps élevé (3.15 secondes).
- **Plusieurs threads :** Amélioration significative (2.39s avec 2 threads, 1.86s avec 4 threads, 2.16s avec 8 threads). Le meilleur résultat est obtenu avec 4 threads.

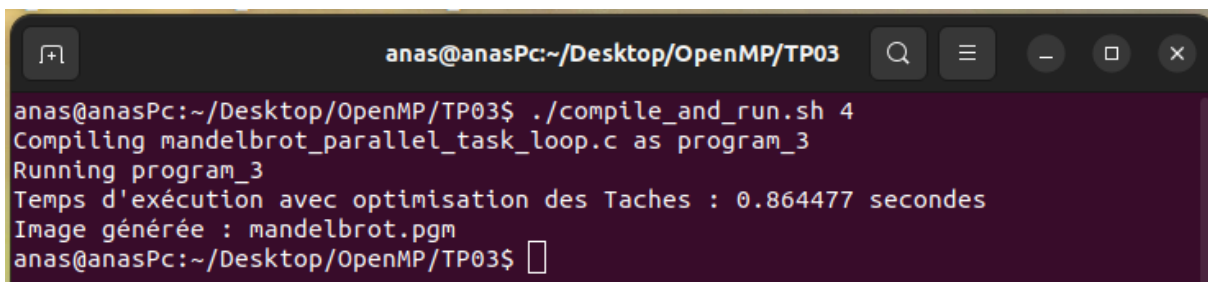
3.4.2 Gestion des Tâches :

- **Surcharge :** Créer une tâche pour chaque pixel peut entraîner une surcharge, ce qui peut affecter les performances.
- **Équilibrage :** La gestion des tâches aide à répartir le travail entre les threads, surtout lorsque les temps de calcul varient.

4 Optimisation des Tâches

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <complex.h>
4 #include <omp.h>
5
6 #define WIDTH 1600
7 #define HEIGHT 1600
8 #define MAX_ITER 1000
9
10 int mandelbrot(double complex c) {
11     double complex z = 0 + 0*I;
12     int iter;
13     for (iter = 0; iter < MAX_ITER; iter++) {
14         if (cabs(z) > 2.0) break;
15         z = z*z + c;
16     }
17     return iter;
18 }
19
20 void generate_mandelbrot(unsigned char image[HEIGHT][WIDTH]) {
21     #pragma omp parallel
22     {
23         #pragma omp single
24         {
25             #pragma omp taskloop grainsize(100)
26             for (int y = 0; y < HEIGHT; y++) {
27                 for (int x = 0; x < WIDTH; x++) {
28                     double complex c = (x - WIDTH / 2.0) * 4.0 /
29                     WIDTH + (y - HEIGHT / 2.0) * 4.0 / HEIGHT * I;
30                     int value = mandelbrot(c);
31                     image[y][x] = (value == MAX_ITER) ? 0 : (255 *
32                     value / MAX_ITER);
33                 }
34             }
35         }
36     }
```

```
34     }
35 }
36
37 int main() {
38     double start, end;
39     unsigned char image[HEIGHT][WIDTH];
40
41     start = omp_get_wtime();
42
43     generate_mandelbrot(image);
44
45     end = omp_get_wtime();
46
47     printf("Temps d'exécution avec optimisation des Taches : %f\n", end - start);
48
49     FILE *fp = fopen("/home/anas/Desktop/mandelbrot.pgm", "wb");
50
51     fprintf(fp, "P5\n%d %d\n255\n", WIDTH, HEIGHT);
52     fwrite(image, 1, WIDTH * HEIGHT, fp);
53     fclose(fp);
54     printf("Image generee : mandelbrot.pgm\n");
55     return 0;
56 }
```

A terminal window titled 'anas@anasPc:~/Desktop/OpenMP/TP03' with standard window controls. The terminal shows the following commands and output:

```
anas@anasPc:~/Desktop/OpenMP/TP03$ ./compile_and_run.sh 4
Compiling mandelbrot_parallel_task_loop.c as program_3
Running program_3
Temps d'exécution avec optimisation des Taches : 0.864477 secondes
Image générée : mandelbrot.pgm
anas@anasPc:~/Desktop/OpenMP/TP03$
```

FIGURE 4 – Temps d'exécution parallèle avec optimisation des Tâches