

UNIVERSITÉ IBN TOFAIL

FACULTÉ DES SCIENCES - KENITRA

DÉPARTEMENT INFORMATIQUE

MASTER INFORMATIQUE ET INTELLIGENCE ARTIFICIELLE

**COMPTE RENDU**

---

## **TP 2 : Parallélisation de la Multiplication de Matrices avec OpenMP**

---

*Réalisé par :*

Anas BOUKHLIJA

*Encadré par :*

Pr. Nada FAQIR

ANNÉE UNIVERSITAIRE 2024-2025

# Table des matières

1	Implémentation Séquentielle de la Multiplication de Matrices . . . . .	3
2	Parallélisation de la Multiplication de Matrices . . . . .	4
3	Analyse des Performances . . . . .	6
3.1	Temps d'exécution séquentiel . . . . .	6
3.2	Temps d'exécution parallèle . . . . .	7
3.3	Temps d'exécution parallèle avec différents nombres de threads . . . . .	9
4	Parallélisation Avancée . . . . .	11
5	Optimisation des Données . . . . .	13
6	Utilisation de Matrices Non Carrées . . . . .	15

# Table des figures

1	Temps d'exécution séquentiel . . . . .	7
2	Temps d'exécution parallèle . . . . .	8
3	Temps d'exécution parallèle avec différents nombres de threads . . . . .	10
4	Parallélisation Avancée : Parallélisation de la boucle interne . . . . .	12
5	Optimisation des Données . . . . .	14
6	Utilisation de Matrices Non Carrées . . . . .	16

# 1 Implémentation Séquentielle de la Multiplication de Matrices

- Écrire un programme C qui effectue la multiplication de deux matrices carrées séquentiellement.
- Initialiser les matrices avec des valeurs aléatoires ou constantes.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define N 500
5
6  int main() {
7
8      int matriceA[N][N];
9      int matriceB[N][N];
10     int matriceResultat[N][N] = {0};
11
12     // Initialisation avec des valeurs aleatoire entre 0 et 10
13     for (int i = 0; i < N; i++) {
14         for (int j = 0; j < N; j++) {
15             matriceA[i][j] = rand() % 10;
16             matriceB[i][j] = rand() % 10;
17         }
18     }
19
20     // Multiplication
21     for (int i = 0; i < N; i++) {
22         for (int j = 0; j < N; j++) {
23             matriceResultat[i][j] = 0;
24             for (int k = 0; k < N; k++) {
25                 matriceResultat[i][j] += matriceA[i][k] *
matriceB[k][j];
26             }
27         }
28     }
29 }
```

```
30
31 // Resultat
32 for (int i = 0; i < N; i++) {
33     for (int j = 0; j < N; j++) {
34         printf("%d\t", matriceResultat[i][j]);
35     }
36     printf("\n");
37 }
38 return 0;
39 }
```

## 2 Parallélisation de la Multiplication de Matrices

- Utilisez OpenMP pour paralléliser les boucles de multiplication des matrices.
- Ajouter les directives OpenMP appropriées pour paralléliser les boucles externes.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define N 500
5
6  int main() {
7
8      int matriceA[N][N];
9      int matriceB[N][N];
10     int matriceResultat[N][N] = {0};
11
12     // Initialisation
13     #pragma omp parallel for collapse(2)
14     for (int i = 0; i < N; i++) {
15         for (int j = 0; j < N; j++) {
16             // Valeur aleatoire entre 0 et 9
17             matriceA[i][j] = rand() % 10;
18             matriceB[i][j] = rand() % 10;
19         }
20     }
21 }
```

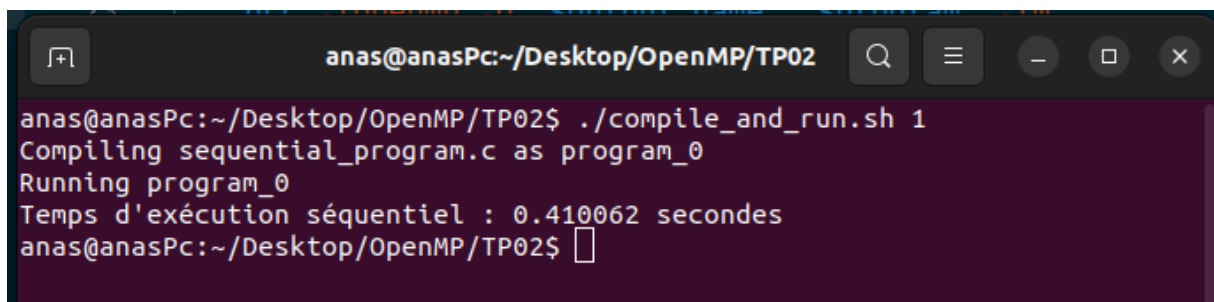
```
22 // Multiplication
23 #pragma omp parallel for collapse(2)
24 for (int i = 0; i < N; i++) {
25     for (int j = 0; j < N; j++) {
26         matriceResultat[i][j] = 0;
27         for (int k = 0; k < N; k++) {
28             matriceResultat[i][j] += matriceA[i][k] * matriceB
[k][j];
29         }
30     }
31 }
32
33 // Resultat
34 #pragma omp parallel for collapse(2)
35 for (int i = 0; i < N; i++) {
36     for (int j = 0; j < N; j++) {
37         printf("%d\t", matriceResultat[i][j]);
38     }
39     printf("\n");
40 }
41
42
43 return 0;
44 }
```

## 3 Analyse des Performances

### 3.1 Temps d'exécution séquentiel

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #include <omp.h>
5
6 #define N 500
7
8 int main() {
9     int matriceA[N][N], matriceB[N][N], matriceResultat[N][N] =
10     {0};
11     double start, end;
12
13     // Initialisation des matrices
14     for (int i = 0; i < N; i++) {
15         for (int j = 0; j < N; j++) {
16             matriceA[i][j] = rand() % 10;
17             matriceB[i][j] = rand() % 10;
18         }
19     }
20
21     // Mesure du temps de multiplication sequentielle
22     start = omp_get_wtime();
23
24     for (int i = 0; i < N; i++) {
25         for (int j = 0; j < N; j++) {
26             matriceResultat[i][j] = 0;
27             for (int k = 0; k < N; k++) {
28                 matriceResultat[i][j] += matriceA[i][k] * matriceB
29                 [k][j];
30             }
31         }
32     }
```

```
32     end = omp_get_wtime();
33     printf("Temps d'exécution séquentiel : %f secondes\n", end -
34           start);
35
36     return 0;
37 }
```



```
anas@anasPc:~/Desktop/OpenMP/TP02$ ./compile_and_run.sh 1
Compiling sequential_program.c as program_0
Running program_0
Temps d'exécution séquentiel : 0.410062 secondes
anas@anasPc:~/Desktop/OpenMP/TP02$
```

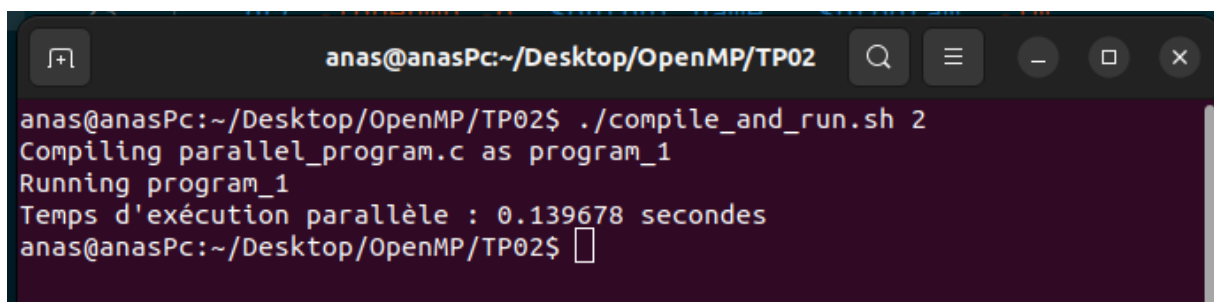
FIGURE 1 – Temps d'exécution séquentiel

### 3.2 Temps d'exécution parallèle

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #include <omp.h>
5
6 #define N 500
7
8 int main() {
9     int matriceA[N][N], matriceB[N][N], matriceResultat[N][N] =
10     {0};
11     double start, end;
12
13     // Initialisation des matrices
14     for (int i = 0; i < N; i++) {
15         for (int j = 0; j < N; j++) {
16             matriceA[i][j] = rand() % 10;
17             matriceB[i][j] = rand() % 10;
18         }
19     }
```



```
19
20 // Mesure du temps de multiplication parallele
21 start = omp_get_wtime();
22
23 #pragma omp parallel for collapse(2)
24 for (int i = 0; i < N; i++) {
25     for (int j = 0; j < N; j++) {
26         matriceResultat[i][j] = 0;
27         for (int k = 0; k < N; k++) {
28             matriceResultat[i][j] += matriceA[i][k] * matriceB
[k][j];
29         }
30     }
31 }
32
33 end = omp_get_wtime();
34 printf("Temps d'execution parallele : %f secondes\n", end -
start);
35
36 return 0;
37 }
```



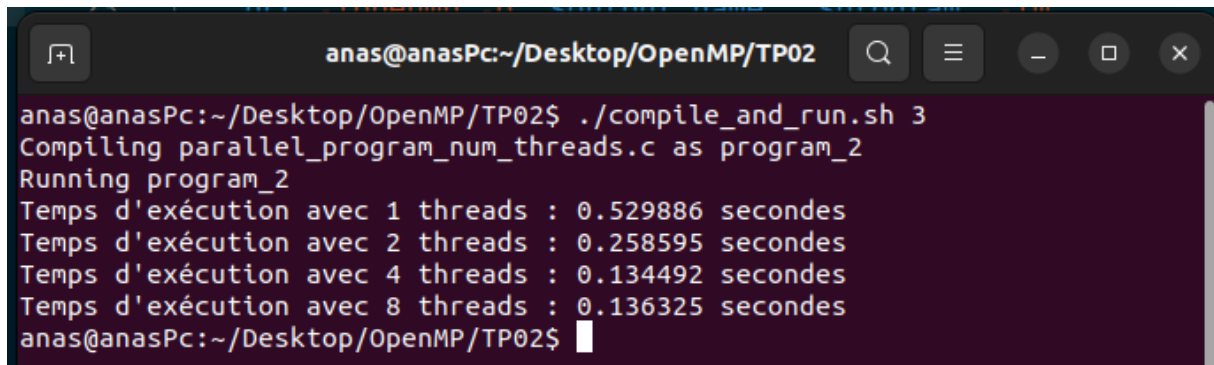
```
anas@anasPc:~/Desktop/OpenMP/TP02$ ./compile_and_run.sh 2
Compiling parallel_program.c as program_1
Running program_1
Temps d'execution parallèle : 0.139678 secondes
anas@anasPc:~/Desktop/OpenMP/TP02$
```

FIGURE 2 – Temps d'exécution parallèle

### 3.3 Temps d'exécution parallèle avec différents nombres de threads

```
1 #include <omp.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <time.h>
5
6 #define N 500
7
8 int main() {
9     int matriceA[N][N], matriceB[N][N], matriceResultat[N][N] =
10     {0};
11     double start, end;
12     int num_threads[] = {1, 2, 4, 8}; // Nombre de threads a
13     tester
14
15     // Initialisation des matrices
16     for (int i = 0; i < N; i++) {
17         for (int j = 0; j < N; j++) {
18             matriceA[i][j] = rand() % 10;
19             matriceB[i][j] = rand() % 10;
20         }
21     }
22
23     for (int t = 0; t < 4; t++) {
24         omp_set_num_threads(num_threads[t]);
25
26         // Mesure du temps de multiplication parallele
27         start = omp_get_wtime();
28
29         #pragma omp parallel for collapse(2)
30         for (int i = 0; i < N; i++) {
31             for (int j = 0; j < N; j++) {
32                 matriceResultat[i][j] = 0;
33                 for (int k = 0; k < N; k++) {
34                     matriceResultat[i][j] += matriceA[i][k] *
35                     matriceB[k][j];
36                 }
37             }
38         }
39     }
40     end = omp_get_wtime();
41     printf("Temps d'exécution : %f\n", end - start);
42 }
```

```
34     }  
35 }  
36  
37     end = omp_get_wtime();  
38     printf("Temps d'exécution avec %d threads : %f secondes\n"  
39 , num_threads[t], end - start);  
40 }  
41  
42 return 0;  
}
```



The image shows a terminal window titled 'anas@anasPc:~/Desktop/OpenMP/TP02'. The user has executed the command './compile\_and\_run.sh 3'. The output shows the compilation of 'parallel\_program\_num\_threads.c' as 'program\_2' and then the execution of 'program\_2'. The execution results are as follows:

Threads	Execution Time (seconds)
1	0.529886
2	0.258595
4	0.134492
8	0.136325

The terminal prompt is 'anas@anasPc:~/Desktop/OpenMP/TP02\$'.

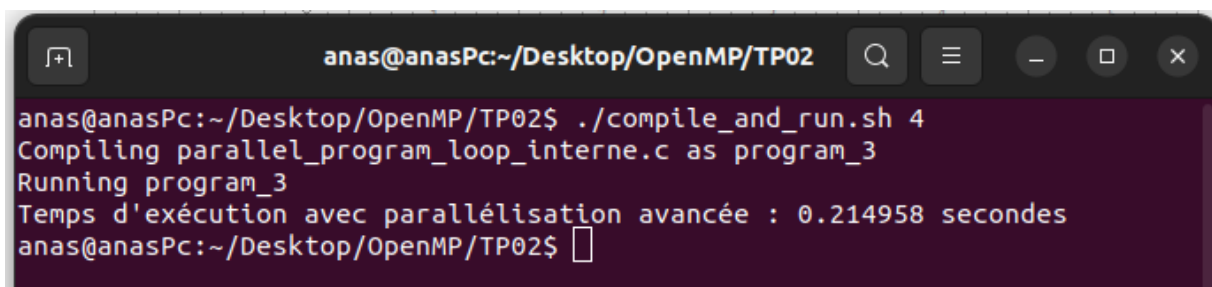
FIGURE 3 – Temps d'exécution parallèle avec différents nombres de threads

## 4 Parallélisation Avancée

Parallélisation de la boucle interne

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #include <omp.h>
5
6 #define N 500
7
8 int main() {
9     int matriceA[N][N], matriceB[N][N], matriceResultat[N][N] =
10     {0};
11     double start, end;
12
13     // Initialisation des matrices
14     for (int i = 0; i < N; i++) {
15         for (int j = 0; j < N; j++) {
16             matriceA[i][j] = rand() % 10;
17             matriceB[i][j] = rand() % 10;
18         }
19     }
20
21     // Mesure du temps de multiplication parallele
22     start = omp_get_wtime();
23
24     #pragma omp parallel for collapse(3)
25     for (int i = 0; i < N; i++) {
26         for (int j = 0; j < N; j++) {
27             for (int k = 0; k < N; k++) {
28                 #pragma omp atomic
29                 matriceResultat[i][j] += matriceA[i][k] * matriceB
30                 [k][j];
31             }
32         }
33     }
```

```
33     end = omp_get_wtime();  
34     printf("Temps d'exécution avec parallelisation avancée : %f  
secondes\n", end - start);  
35  
36     return 0;  
37 }
```



The image shows a terminal window titled 'anas@anasPc:~/Desktop/OpenMP/TP02'. The user has executed the command './compile\_and\_run.sh 4'. The terminal output shows the compilation of 'parallel\_program\_loop\_interne.c' as 'program\_3', followed by the execution of 'program\_3'. The output of the program is 'Temps d'exécution avec parallélisation avancée : 0.214958 secondes'. The prompt 'anas@anasPc:~/Desktop/OpenMP/TP02\$' is visible at the bottom.

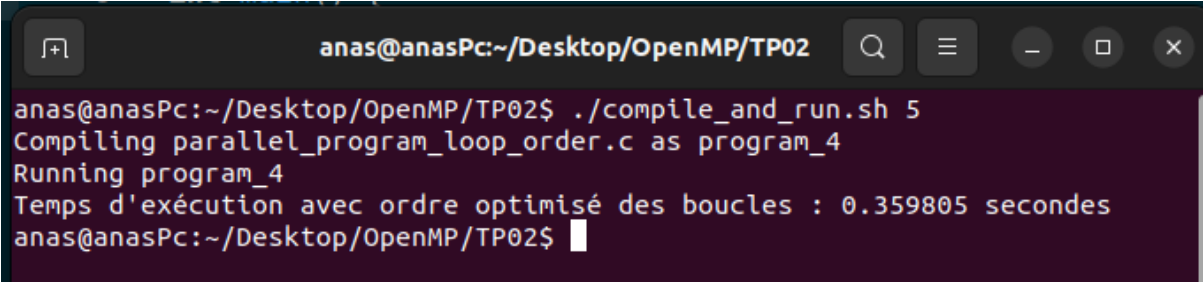
```
anas@anasPc:~/Desktop/OpenMP/TP02$ ./compile_and_run.sh 4  
Compiling parallel_program_loop_interne.c as program_3  
Running program_3  
Temps d'exécution avec parallélisation avancée : 0.214958 secondes  
anas@anasPc:~/Desktop/OpenMP/TP02$
```

FIGURE 4 – Parallélisation Avancée : Parallélisation de la boucle interne

## 5 Optimisation des Données

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <omp.h>
4
5 #define N 500
6
7 int main() {
8     int matriceA[N][N], matriceB[N][N], matriceResultat[N][N] =
9     {0};
10
11     double start, end;
12
13     // Initialisation des matrices
14     for (int i = 0; i < N; i++) {
15         for (int j = 0; j < N; j++) {
16             matriceA[i][j] = rand() % 10;
17             matriceB[i][j] = rand() % 10;
18         }
19     }
20
21     // Mesure du temps d'exécution avec acces memoire optimise
22     start = omp_get_wtime();
23
24     // Ordre optimise des boucles
25     for (int i = 0; i < N; i++) {
26         for (int k = 0; k < N; k++) {
27             for (int j = 0; j < N; j++) {
28                 matriceResultat[i][j] += matriceA[i][k] * matriceB
29                 [k][j];
30             }
31         }
32     }
33
34     end = omp_get_wtime();
35     printf("Temps d'exécution avec ordre optimise des boucles : %f
36     secondes\n", end - start);
```

```
33  
34     return 0;  
35 }
```



The terminal window shows the execution of a script `./compile_and_run.sh 5`. The output indicates that the program `parallel_program_loop_order.c` was compiled as `program_4` and then executed. The execution time with optimized loop order is reported as `0.359805 secondes`.

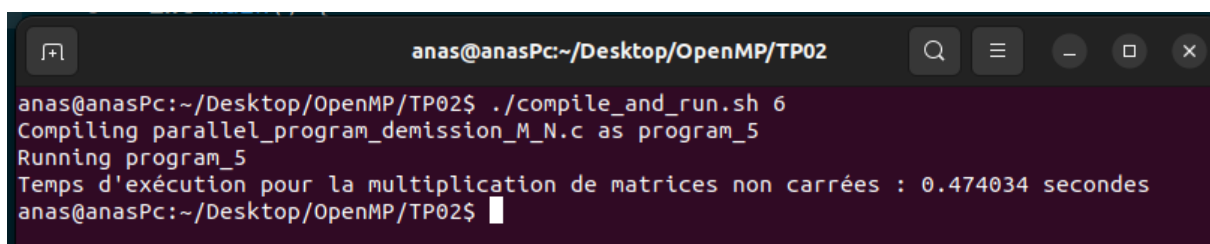
FIGURE 5 – Optimisation des Données

## 6 Utilisation de Matrices Non Carrées

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <omp.h>
4
5 #define ROWS_A 500
6 #define COLS_A 500
7 #define COLS_B 500
8
9 int main() {
10     int matriceA[ROWS_A][COLS_A], matriceB[COLS_A][COLS_B],
11     matriceResultat[ROWS_A][COLS_B];
12     double start, end;
13
14     // Initialisation des matrices
15     for (int i = 0; i < ROWS_A; i++) {
16         for (int j = 0; j < COLS_A; j++) {
17             matriceA[i][j] = rand() % 10;
18         }
19     }
20
21     for (int i = 0; i < COLS_A; i++) {
22         for (int j = 0; j < COLS_B; j++) {
23             matriceB[i][j] = rand() % 10;
24         }
25     }
26
27     // Initialiser matriceResultat a zero
28     for (int i = 0; i < ROWS_A; i++) {
29         for (int j = 0; j < COLS_B; j++) {
30             matriceResultat[i][j] = 0;
31         }
32     }
33
34     // Mesure du temps d'execution pour la multiplication
35     start = omp_get_wtime();
```



```
35
36 // Multiplication des matrices
37 for (int i = 0; i < ROWS_A; i++) {
38     for (int k = 0; k < COLS_A; k++) {
39         for (int j = 0; j < COLS_B; j++) {
40             matriceResultat[i][j] += matriceA[i][k] * matriceB
41             [k][j];
42         }
43     }
44
45     end = omp_get_wtime();
46     printf("Temps d'exécution pour la multiplication de matrices
47     non carrees : %f secondes\n", end - start);
48
49     return 0;
50 }
```



```
anas@anasPc:~/Desktop/OpenMP/TP02$ ./compile_and_run.sh 6
Compiling parallel_program_démission_M_N.c as program_5
Running program_5
Temps d'exécution pour la multiplication de matrices non carrées : 0.474034 secondes
anas@anasPc:~/Desktop/OpenMP/TP02$
```

FIGURE 6 – Utilisation de Matrices Non Carrées