OS202 - Compte rendu du Projet

Arthur Bouvet et Diana Tymoshenko

March 2025

Table des matières

1 Présentation de la machine

Voici la configuration de l'ordinateur sur lequel j'ai effectué mon projet :

```
Architecture:            x86_64
CPU op-mode(s):          32-bit, 64-bit
Address sizes:           39 bits physic
                         al, 48 bits vi
                         rtual
Byte Order:              Little Endian
CPU(s):                  16
On-line CPU(s) list:     0-15
Vendor ID:               GenuineIntel
Model name:              13th Gen Intel
                         (R) Core(TM) i
                         7-1360P
CPU family:              6
Model:                   186
Thread(s) per core:      2
Core(s) per socket:      8
Socket(s):               1
Stepping:                2
BogoMIPS:                5222.36
```

Nous travaillons donc avec 16 coeurs physiques, en natif (via WSL Ubuntu).

2 Partie 1

2.1 Mesures de temps

La première étape de ce travail consiste à analyser le temps mis par le processeur pour itérer sur la simulation du feu de forêt. En particulier, nous cherchons à étudier l'influence du calcul (donc du lancement de la simulation à proprement dit) et de l'affichage de l'avancée du feu à chaque itération.

Le mode opératoire retenu est le suivant : dans le main du fichier *simulation.cpp*, nous mesurons (grâce à des différences de chronomètres) à chaque itération : le temps pris par l'appel à

la fonction *simu.update*, le temps pris par l'appel à *displayer->update* et le temps total qui est la somme des 2 précédents. Nous ajoutons ces valeurs dans 3 vecteurs contenant les temps pour chaque itération. Une fois les itérations terminées, nous faisons appel à la fonction *print_statistics* qui calcule pour les 3 mesures qui nous intéressent, le temps total que représente l'étape dans toute la simulation, le temps minimum pris par cette étape au cours d'une itération ainsi que le temps maximum.

```
Parametres définis pour la simulation :
        Taille du terrain : 1
        Nombre de cellules par direction : 1000
        Vecteur vitesse : [0, 0]
        Position initiale du foyer (col, ligne) : 500, 500

==== Itération complète ====
Nombre de mesures: 500
Temps moyen: 82.4119 ms
Temps min: 24.167 ms
Temps max: 188.133 ms

==== Mise à jour du modèle ====
Nombre de mesures: 500
Temps moyen: 14.2489 ms
Temps min: 0.007 ms
Temps max: 39.058 ms

==== Affichage ====
Nombre de mesures: 500
Temps moyen: 68.1622 ms
Temps min: 24.041 ms
Temps max: 151.13 ms

==== Itération complète ====
Nombre de mesures: 500
Temps moyen: 127.385 ms
Temps min: 23.848 ms
```

```
Temps max: 255.175 ms

==== Mise à jour du modèle ====
Nombre de mesures: 500
Temps moyen: 22.2579 ms
Temps min: 0.008 ms
Temps max: 54.263 ms

==== Affichage ====
Nombre de mesures: 500
Temps moyen: 105.126 ms
Temps min: 23.453 ms
Temps max: 208.497 ms

==== Itération complète ====
Nombre de mesures: 500
Temps moyen: 85.3391 ms
Temps min: 24.24 ms
Temps max: 222.209 ms

==== Mise à jour du modèle ====
Nombre de mesures: 500
Temps moyen: 15.6003 ms
Temps min: 0.008 ms
Temps max: 45.553 ms

==== Affichage ====
Nombre de mesures: 500
Temps moyen: 69.7379 ms
Temps min: 23.944 ms
Temps max: 177.404 ms
```

3 Partie 2

— Repartez de la version originale du code.
— Mettre en place l'environnement MPI dans votre code.
— Séparez l'affichage qui sera fait par le processus n°0 du calcul qui sera fait par les processus de numéro non nuls.
— Testez le code avec deux processus et mesurez le temps global moyen pris par itération en temps.
— Interprétez votre résultat par rapport au temps global mesuré sur le code d'origine.

3.1 Configuration de la machine

La tâche a été effectuée sur un ordinateur avec la configuration suivante.

```
Architecture:            x86_64
CPU op-mode(s):          32-bit, 64-bit
Address sizes:           39 bits physical,
                         48 bits virtual
Byte Order:              Little Endian
CPU(s):                  8
On-line CPU(s) list:     0-7
Vendor ID:               GenuineIntel
Model name:              Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
CPU family:              6
Model:                   142
Thread(s) per core:      2
Core(s) per socket:      4
Socket(s):               1
Stepping:                10
BogoMIPS:                3600.00

Caches (sum of all):
L1d:                     128 KiB (4 instances)
L1i:                     128 KiB (4 instances)
L2:                      1 MiB (4 instances)
L3:                      6 MiB (1 instance)
```

Nous travaillons avec 4 coeurs physiques, 8 coeurs logiques en natif (Ubuntu).

3.2 Mesures de temps

Le programme s'exécute avec les paramètres suivants.

```
Parametres définis pour la simulation :
        Taille du terrain : 1
        Nombre de cellules par direction : 1000
        Vecteur vitesse : [0, 0]
        Position initiale du foyer (col, ligne) : 500, 500
```

Pour calculer le temps, nous effectuons 500 itérations, enregistrons le temps à chaque itération, puis calculons le temps moyen.

Messures	Code d'origine	MPI (nbp = 2)	Acceleration
Temps moyen	117.141 ms	65.8487 ms	1.78
Temps min	68.103 ms	62.713 ms	-
Temps max	242.431 ms	149.112 ms	-

TABLE 1 – Temps d'itération complète

Messures	Code d'origine	MPI (nbp = 2)	Acceleration
Temps moyen	41.5922 ms	21.5251 ms	1.932
Temps min	0.005 ms	0.006 ms	-
Temps max	91.032 ms	62.281 ms	-

TABLE 2 – Temps de l'avancement en temps

Comme on peut le voir dans les tableaux, l'exécution parallèle permet d'accélérer presque deux fois le temps d'une itération complète ainsi que le temps des calculs.

En ce qui concerne le temps d'une itération complète, la réduction est due au fait que l'affichage et le calcul des données peuvent se dérouler en parallèle, chacun étant réalisé par un processus distinct. Cependant, l'affichage prend plus de temps que le calcul. Dans un programme utilisant MPI, un temps supplémentaire est ajouté en raison de la communication entre les processus.

Quant au temps moyen des calculs, il a également diminué, bien que, comme dans le programme séquentiel, un seul processus effectue les calculs. Cela pourrait être lié à l'utilisation du cache, puisque le processus concerné ne travaille que sur les données de calcul et ne s'occupe pas de l'affichage des résultats. Chaque processus est affecté à un cœur spécifique, ce qui peut améliorer la localité du cache et réduire les commutations de contexte.

4 Partie 4

— Définissez un groupe de communication pour les processus impliqués dans le calcul de la simulation.
— Découpez la zone en tranche en fonction du nombre de processus impliqués dans le calcul.
— Parallélisez le calcul à partir de ce découpage en tranche et en utilisant des cellules fantômes.
— Calculez l'accélération globale ainsi que l'accélération de l'avancement en temps.
— Comparez les résultats obtenus avec les résultats obtenus dans le code écrit pour la troisième étape.
— En analysant vos résultats, suggérez des méthodes afin d'améliorer l'accélération obtenue avec MPI.

4.1 Mesures de temps

La configuration de l'ordnateur a été présentée dans la partie 2.

Dans ce programme, nous divisons le modèle en blocs par lignes en fonction du nombre de processus, chaque processus calcule sa propre partie, et les processus échangent entre eux des cellules fantômes.

nbp	Temps moyen	Temps min	Temps max	Acceleration
3	65.2361 ms	61.941 ms	177.199 ms	1.79
4	66.2127 ms	61.731 ms	216.341 ms	1.76
6	195.171 ms	63.19 ms	395.906 ms	0.6
8	230.726 ms	159.089 ms	386.862 ms	0.507

TABLE 3 – Temps d'itération complète

nbp	Temps moyen	Temps min	Temps max	Acceleration
3	13.6634 ms	0.018 ms	38.953 ms	3.04
4	18.2233 ms	0.022 ms	45.042 ms	2.28
6	21.0896 ms	0.023 ms	42.305 ms	1.97
8	23.1166 ms	0.018 ms	60.433 ms	1.79

TABLE 4 – Temps de l'avancement en temps

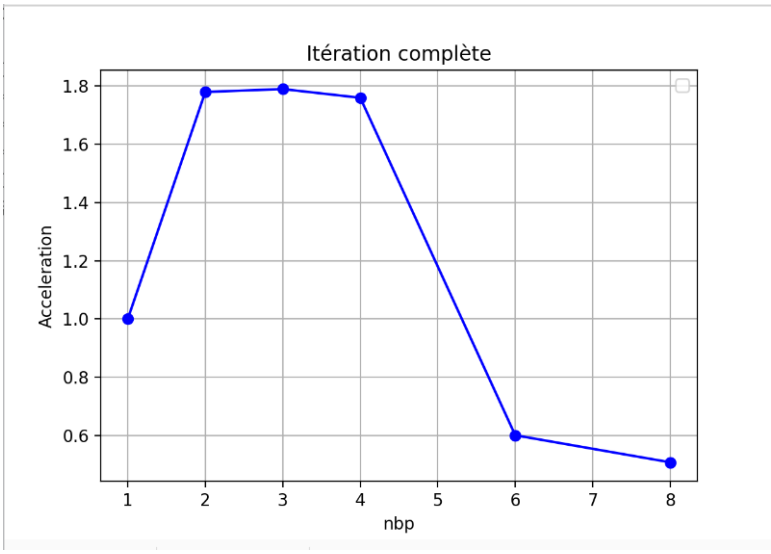

FIGURE 1 – Courbe : itération complète

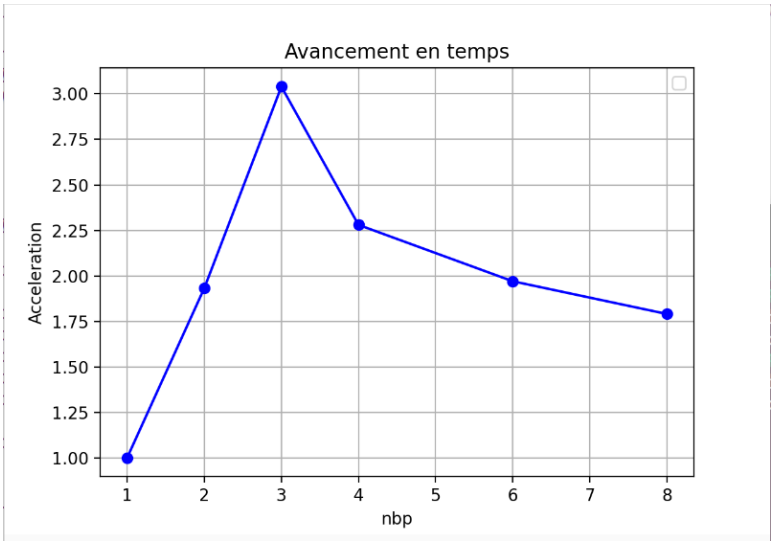

FIGURE 2 – Courbe : avancement en temps

Comme on le voit dans les résultats, il y a une accélération, mais davantage de processus pour les calculs nécessitent plus de communication entre eux, car il y a un échange de cellules fantômes, ce qui peut augmenter le temps. De plus, le processus 1, qui est responsable en plus des calculs de l'interaction avec le processus 0, peut ralentir les autres processus, car dans certains endroits, il y a une synchronisation.