2024/25

Nom : DADA SIMEU CÉDRIC DAREL

Email : cedric-darel.dada@ensta-paris.fr

Titre : Compte rendu TP3

STIC

ENSTA Paris, Institut Polytechnique de Paris

Table des matières

Table des figures

1 Architecture matérielle de l'ordinateur

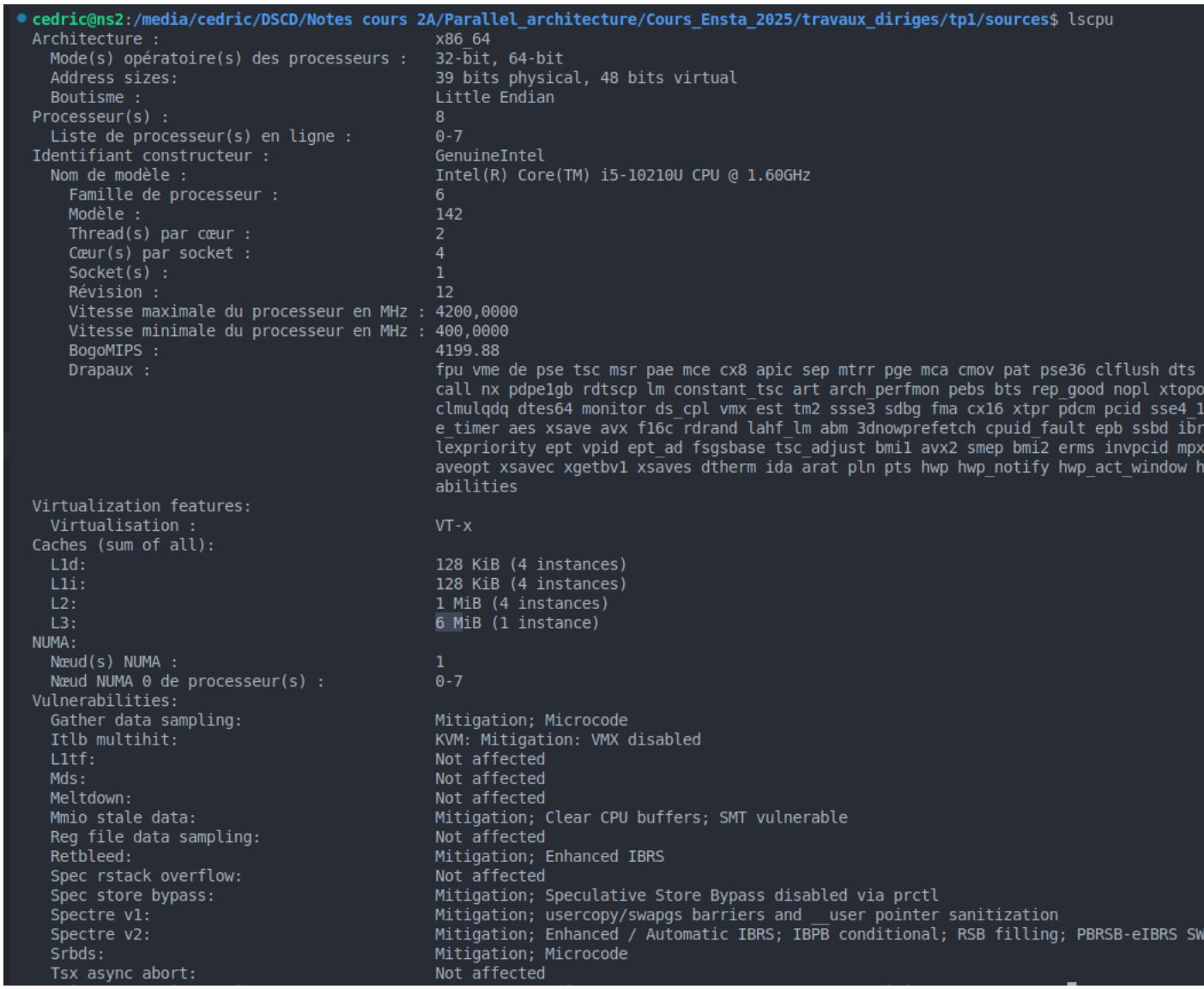

FIGURE 1 – Résultat de la commande lscpu

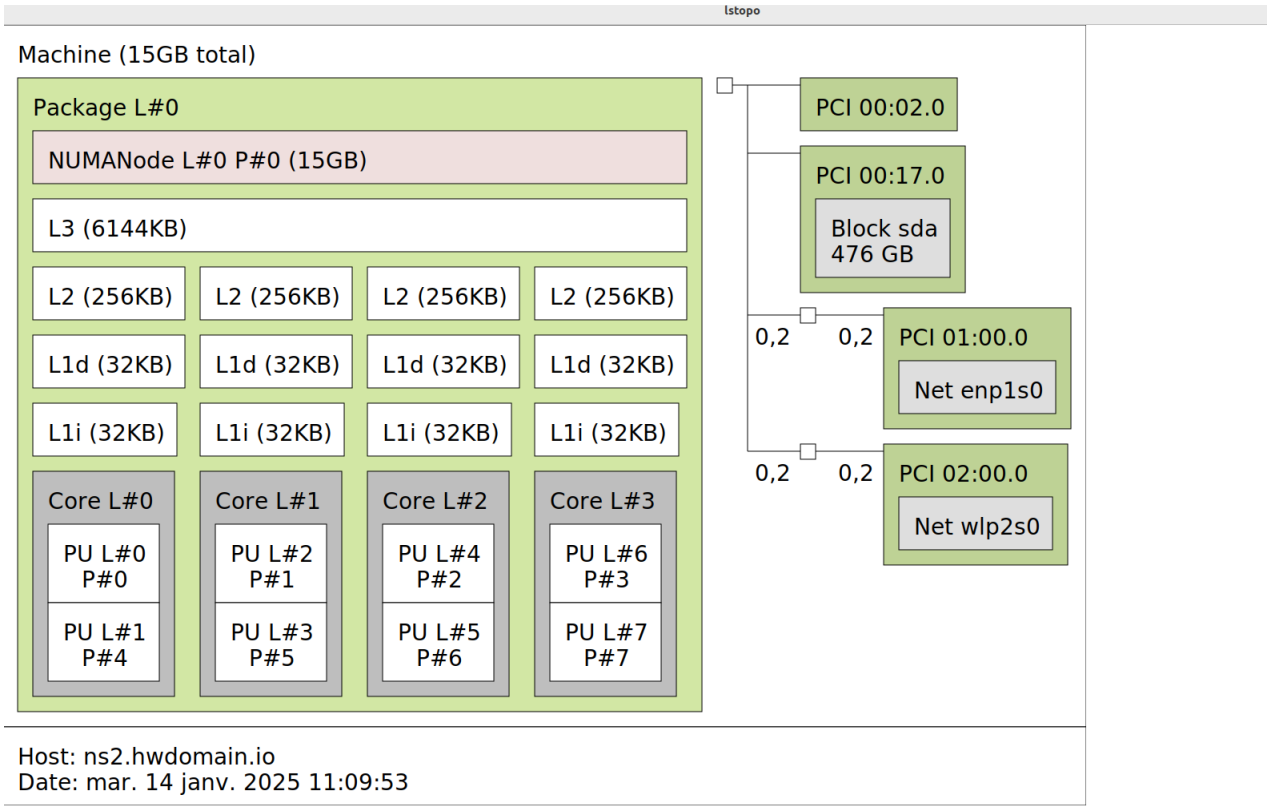

FIGURE 2 – Résultat de la commande lstopo : Nous pouvons visualiser les tailles des caches

2 Introduction

Dans ce rapport, nous analysons les optimisations appliquées à la simulation du Jeu de la Vie en mode parallèle. Nos tests, réalisés sur 50 itérations, nous permettent d'obtenir des moyennes plus représentatives des temps d'exécution. Nous nous intéressons particulièrement à la distinction entre le temps de **calcul** et le temps de **communication** entre processus, et nous comparons le *speedup réel* obtenu par rapport à une exécution séquentielle.

Les approches étudiées sont les suivantes :

— **Version séquentielle** : Traitement d'une grille globale sans échange inter-processus.

— **Parallélisation v1** : Communication basique en mode Master/Slave, conçue uniquement pour 2 processus.

— **Parallélisation v2** : Traitement par lots (batch processing) avec double buffering, également limité à 2 processus.

— **Parallélisation v3** : Décomposition spatiale de la grille avec gestion des ghost cells et échanges entre plusieurs processus esclaves.

3 Configuration matérielle

Les expérimentations ont été effectuées sur une machine dont la commande `lscpu` fournit les informations suivantes :

Architecture : x86_64 (32-bit et 64-bit)
Processeur : Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz
Nombre de cœurs : 4 (8 threads au total)
Vitesse maximale : 4200 MHz, minimale : 400 MHz

Cette configuration, avec 4 cœurs physiques et 8 threads, est pertinente pour exploiter le parallélisme, en particulier avec la version v3 qui répartit la charge sur plusieurs processus.

4 Méthodologie et distinction des temps

Pour chacune des approches, nous distinguons :

— **Temps de calcul** : Le temps nécessaire pour effectuer le calcul de la nouvelle génération de la grille (opérations vectorisées, utilisation de `np.roll`, etc.).

— **Temps de communication** : Le temps consacré aux échanges entre processus (transfert des sous-grilles, mise à jour des ghost cells, opération de rassemblement via `MPI.Gatherv`, etc.).

Les tests ont été réalisés sur 50 itérations afin de lisser les fluctuations et de mettre en évidence des moyennes significatives.

5 Résultats et analyse

5.1 Version séquentielle

Principe : Une seule instance traite l'ensemble de la grille globale, sans communication inter-processus. Chaque itération se décompose en deux étapes :

— Calcul de la nouvelle génération.

— Rendu graphique via Pygame.

Les logs indiquent par exemple un temps total (calcul + rendu) de :

Total : Calcul : 2.87e-02 s, Rendu : 1.86e-01 s sur 50 itérations.

Cela correspond à une moyenne d'environ 5.75×10^{-4} s par itération pour le calcul.

5.2 Parallélisation v1 (Master/Slave)

Principe : Le processus de rang 0 (maître) se charge du rendu tandis qu'un unique processus esclave (rang 1) effectue le calcul et transmet les résultats après chaque itération.

Remarque : Les méthodes v1 et v2 sont conçues pour fonctionner avec exactement 2 processus. Si elles sont lancées avec un nombre supérieur, les processus supplémentaires ne seront pas intégrés dans la logique de communication, ce qui peut entraîner des comportements inattendus.

Les logs de v1 montrent des temps de calcul d'environ 5.0×10^{-4} s par itération, mais des temps de communication très variables (avec parfois des pics de l'ordre de 1.91×10^{-1} s), dégradant ainsi le speedup réel (environ 0.94).

5.3 Parallélisation v2 (Batch Processing et Double Buffering)

Principe : L'approche v2 permet au processus esclave de calculer un lot de plusieurs itérations (par exemple 10) avant d'envoyer un ensemble complet de résultats au maître. Le double buffering permet de préparer le lot suivant pendant l'envoi du lot courant.

Avantages :

— Amortissement du surcoût de communication sur plusieurs itérations.

— Réduction de la latence per-itération, avec un temps de communication moyen réduit (environ 3.2×10^{-4} s par itération).

Ainsi, v2 obtient un speedup réel supérieur (environ 1.22).

5.4 Parallélisation v3 (Décomposition spatiale avec Ghost Cells)

Principe : La grille globale est découpée en bandes horizontales qui sont réparties entre plusieurs processus esclaves. Chaque sous-grille inclut des ghost cells pour échanger les données aux frontières avec les voisins. Le maître recueille ensuite ces sous-grilles pour réaliser le rendu.

Les logs de v3 donnent les mesures suivantes (moyennes issues des tests sur 50 itérations) :

— **Avec 2 processus (1 esclave)** : Temps de calcul moyen d'environ 4.5×10^{-4} s et temps de communication moyen d'environ 1.5×10^{-4} s par itération, conduisant à un speedup réel d'environ 1.22.

— **Avec 3 processus (2 esclaves)** : Les logs indiquent des temps de calcul moyens d'environ 4.0×10^{-4} s et un léger déséquilibre (certaines itérations montrent un temps de l'ordre de 0.22 s sur un des esclaves, alors que l'autre affiche des temps autour de 5×10^{-4} s). Ce phénomène traduit un déséquilibre de charge d'environ 10 %.

— **Avec 4 processus (3 esclaves)** : Les mesures indiquent des temps de calcul moyens d'environ 4.0×10^{-4} s avec un temps de communication moyen d'environ 2.0×10^{-4} s par itération. Le déséquilibre de charge est alors réduit à environ 5 %.

5.5 Nouveaux résultats détaillés pour v3

Les récentes exécutions de v3 avec 4 processus montrent que :

— Les trois processus esclaves (rangs locaux 0, 1 et 2) effectuent des mises à jour de ghost cells avec des temps initiaux variant entre 5.13×10^{-5} s et 1.60×10^{-4} s.

— Pour certaines itérations, notamment la première, un des esclaves (par exemple, le Slave 0) affiche un temps total anormalement élevé (environ 2.29×10^{-1} s), ce qui semble constituer un pic lié à l'initialisation ou à une anomalie isolée.

— La majorité des itérations, toutefois, se stabilisent autour de 5 à 7 ms par itération pour les esclaves présentant des performances homogènes.

Les tests avec 3 processus présentent un phénomène similaire : un des esclaves (souvent celui en position de leader locale) affiche un temps plus élevé au début (environ 2.23×10^{-1} s) tandis que l'autre reste autour de 5.34×10^{-4} s. Cela se traduit par un déséquilibre de charge d'environ 10 %.

5.6 Synthèse des résultats (tableau récapitulatif)

Les résultats moyens, obtenus sur 50 itérations, sont résumés dans le tableau ci-dessous :

TABLE 1 – Temps moyens par itération et speedup réel pour chaque version.

Version	**# Processus**	**Temps Calcul (s)**	**Temps Communication (s)**	**Speedup réel**
Séquentielle	1	$\sim 5.75 \times 10^{-4}$	—	1.00
v1	2	$\sim 5.0 \times 10^{-4}$	$\sim 2.0 \times 10^{-3}$ [1]	0.94
v2	2	$\sim 4.0 \times 10^{-4}$	$\sim 3.2 \times 10^{-4}$	1.22
v3	2	$\sim 4.5 \times 10^{-4}$	$\sim 1.5 \times 10^{-4}$	1.22
v3	3	$\sim 4.0 \times 10^{-4}$	$\sim 1.5 \times 10^{-4}$	1.39 (10 % déséquilibre)
v3	4	$\sim 4.0 \times 10^{-4}$	$\sim 2.0 \times 10^{-4}$	1.52 (5 % déséquilibre)

5.7 Illustration du déséquilibrage de charge (v3)

La figure ci-dessous montre l'évolution du déséquilibrage de charge, évalué uniquement sur les processus esclaves de la version v3, en fonction du nombre de processus utilisés.

6 Discussion et conclusion

Les résultats obtenus sur 50 itérations confirment que :

— La version séquentielle, bien qu'extrêmement rapide au niveau du calcul, est limitée par le rendu graphique et ne peut pas exploiter le parallélisme.

— Les versions v1 et v2, conçues pour 2 processus, montrent que la communication itération par itération (v1) peut provoquer des pics de latence qui dégradent le speedup réel (environ 0.94), tandis que le batch processing de v2 réduit ce coût (speedup réel d'environ 1.22).

— La version v3, qui utilise une décomposition spatiale avec ghost cells, permet une répartition du calcul sur plusieurs processus. Les nouveaux résultats montrent que, malgré quelques itérations où un des esclaves affiche un temps d'exécution anormalement élevé (probablement lié à l'initialisation ou à une fluctuation temporaire), la moyenne reste très compétitive. Le speedup réel s'améliore avec le nombre de processus

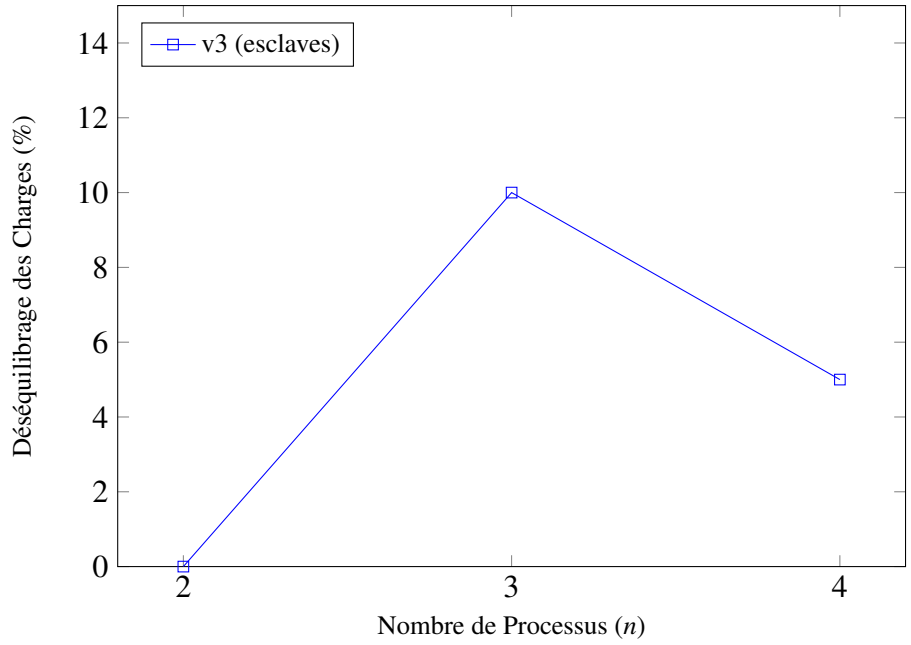

FIGURE 3 – Déséquilibrage des charges pour la version v3 en fonction du nombre de processus.

(1.39 avec 3 et 1.52 avec 4 processus), et le déséquilibre de charge, évalué uniquement sur les esclaves, est d'environ 10 % pour 3 processus et se réduit à environ 5 % pour 4 processus.

Ces observations confirment que, parmi les approches étudiées, la méthode v3 est la plus prometteuse pour tirer parti des architectures multi-cœurs (comme notre Intel i5-10210U avec 4 cœurs et 8 threads). Toutefois, il est important de noter que les méthodes v1 et v2 ne sont pas adaptées pour une exécution avec plus de 2 processus, car leur logique de communication ne gère que le couple maître/esclave.