



Architecture des microprocesseurs (ECE\_4ES01\_TA)

## **TD/TP5 : Analyse de performances de multicoeurs**

Microprocesseur multicoeurs

**Auteurs (quadrinôme)**

CHACÓN GÓMEZ José Daniel

DE MACENA BARRETO Prénom 2

EMMANUEL DA COSTA Lucas

ZHE Chen

**Encadrant / Chargé de TD : SIDEM Antoine**

**Responsable de la matière : HAMMAMI Omar**

## Table des matières

---

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Exercice 3 — A compléter</b>	<b>2</b>
2.1	Questions . . . . .	2
2.1.1	Question 1 . . . . .	2
2.1.2	Question 2 - Comparaison de performances . . . . .	2
2.1.3	Question 3 - Localité de références . . . . .	3

# 1 Introduction

## 2 Exercice 3 — A completer

### 2.1 Questions

#### 2.1.1 Question 1

**Énoncé (Q1).** En considérant que chaque thread s'exécute sur un processeur dans une architecture de type multicoeurs à base de bus et 1 niveau de cache (comme décrit Figure 21), décrivez le comportement de la hiérarchie mémoire et de la cohérence des caches pour l'algorithme de multiplication de matrices. On supposera que le thread principal se trouve sur le processeur d'indice 1.

```

1 // (...)
2
3 #pragma omp parallel for
4 for(int x = 0; x < size; x++) {
5     for (int y = 0; y < size; y++) {
6         int64_t tot;
7         for (int m = 0; m < size; m++) {
8             tot += A[x*size + m]*B[m*size + y];
9         }
10        C[x*size + y] = tot;
11    }
12 }
13
14 // (...)

```

Listing 1 – Utilisation de openMP pour multiplication matricial

Dans l'algorithme de multiplication de matrices présenté dans le Listing 1, chaque thread s'exécute sur un processeur dans une architecture multicoeurs. Chaque thread accède à la mémoire pour lire les éléments des matrices A et B, et pour écrire les résultats dans la matrice C. La hiérarchie mémoire dans cette architecture multicoeurs comprend un niveau de cache partagé entre les processeurs. Lorsque les threads accèdent aux éléments des matrices, ils peuvent bénéficier de la localité de références, ce qui signifie que les données récemment utilisées sont susceptibles d'être réutilisées prochainement. Cependant, si plusieurs threads accèdent simultanément à des éléments de matrice qui ne sont pas contigus en mémoire (par exemple, en accédant à des éléments de la matrice B), cela peut entraîner des cache misses, ce qui ralentit les performances, sachant que les accès à la matrice B sont dans l'ordre de la colonne, ce qui n'est pas optimal pour la localité de référence.

#### 2.1.2 Question 2 - Comparaison de performances

**Énoncé (Q2).** Examinez le fichier de déclaration d'un élément de type « processeur superscalaire out-of-order », et présentez sous forme de tableau cinq paramètres configurables

Paramètre	Description	Valeur par défaut
fetchWidth	N <sup>o</sup> d'instr. que peut récupérer par cycle	4
decodeWidth	N <sup>o</sup> d'instr. que peut décoder par cycle	4
issueWidth	N <sup>o</sup> d'instr. que peut émettre par cycle	4
commitWidth	N <sup>o</sup> d'instr. que peut valider par cycle	4
numROBEntries	N <sup>o</sup> d'entrées dans la Reorder Buffer (ROB)	192

TABLE 1 – Paramètres configurables d'un processeur superscalaire out-of-order

de ce type de processeur avec leur valeur par défaut. Choisissez de préférence des paramètres étudiés lors des séances TD/TP précédentes. Le fichier à consulter est le suivant : `$GEM5/src/cpu/o3/O3CPU.py`

### 2.1.3 Question 3 - Localité de références