TP4 : Architecture des microprocesseurs

Júlia Ellen Dias Leite
julia-ellen.dias@ensta-paris.fr

Charbel Chahla
charbel.chahla@ensta.fr

Martin Brun
martin.brun@ensta.fr

Paul-henri DJOKO
paul-henri.djoko@ensta.fr

18 février 2026

Analyse théorique de cohérence de cache

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.

Paramètres de l'architecture multicoeurs

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 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
```

Q3 : Examinez le fichier d'options de la plateforme se.py, puis déterminez et présentez sous forme de tableau les valeurs par défaut des paramètres suivants :

— Cache de données de niveau 1 : associativité, taille du cache, taille de la ligne
— Cache d'instructions de niveau 1 : associativité, taille du cache, taille de la ligne
— Cache unifié de niveau 2 : associativité, taille du cache, taille de la ligne

Le fichier d'options à consulter est le suivant :

```
$GEM5/configs/common/Options.py
```

Architecture multicoeurs avec des processeurs superscalaires in-order (Cortex A7)

Q4 : Déterminez quel est le processeur exécutant toujours le plus grand nombre de cycles. Expliquez pourquoi. Expliquez également pourquoi l'analyse du nombre de cycles sur ce processeur revient à analyser le nombre total de cycles d'exécution de l'application.

Q5 : Pour chaque configuration, quel est le nombre de cycles d'exécution de l'application ? Vous pourrez présenter vos résultats sous forme de graphe 2 axes

Q6 : Déduire le speedup par rapport à la configuration à 1 thread.

Q7 : En utilisant le nombre total d'instructions simulées, déterminez quelle est la valeur maximale de l'IPC pour chaque configuration ?

Q8 : Discussion et interprétation (max. 10 lignes).

Architecture multicoeurs avec des processeurs superscalaires out-of-order (Cortex A15)

Q9 : Pour chaque configuration, quel est le nombre de cycles d'exécution de l'application ? Vous pourrez présenter vos résultats sous forme de graphe 3 axes.

Temps d'exécution simulé (sim_seconds) en fonction du nombre de threads et de la largeur du processeur :

TABLE 1 – Temps d'exécution (sim_seconds) par largeur de processeur et nombre de threads.

Width	1 thread	2 threads	4 threads	8 threads	16 threads
2 voies	0.000129	0.000094	0.000077	0.000069	0.000067
4 voies	0.000129	0.000094	0.000077	0.000069	0.000067
8 voies	0.000129	0.000094	0.000077	0.000069	0.000067

Les résultats montrent que :
— Le temps d'exécution diminue avec l'augmentation du nombre de threads (parallelism).
— Pour un nombre de threads donné, le temps est **identique** quel que soit la largeur du processeur (2, 4 ou 8 voies).
— Cela suggère que l'application est **fortement limitée par l'équilibre entre threads** plutôt que par la largeur du processeur superscalaire.
— Le temps diminue de 0.000129s (1 thread) à 0.000067s (16 threads), soit une réduction de **48%**.
Remarque : Pour obtenir le **nombre de cycles**, il manque la valeur `sim_ticks` (ou `numCycles`) et la période d'horloge. Sans ces données, on ne peut pas convertir `sim_seconds` en cycles. Cependant, le temps simulé est une métrique valide pour comparer les performances relatives.

Q10 : Déduire le speedup par rapport à la configuration à 1 thread.

Le **speedup** est défini comme le rapport du temps de base (1 thread) au temps pour n threads :

$$\text{Speedup}(n) = \frac{T_1}{T_n}$$

TABLE 2 – Speedup par rapport à 1 thread.

Nombre de threads	2	4	8	16
Speedup	1.37	1.68	1.87	1.93

— Le speedup est **sub-linéaire** : il ne croît pas proportionnellement au nombre de threads.
— Avec 2 threads, on obtient 1.37× (au lieu du 2× théorique).
— Avec 16 threads, on obtient 1.93× (au lieu du 16× théorique).
— Cela indique une **contention sur les ressources partagées** (L2 cache, mémoire, pipeline).

Q11 : En utilisant le nombre total d'instructions simulées, déterminez quelle est la valeur maximale de l'IPC pour chaque configuration ?

L'**IPC (Instructions Per Cycle)** mesure le parallélisme au niveau des instructions exécutées par cycle d'horloge.
— L'IPC pour 1 thread est **2.128** pour tous les widths (identique).
— Les données IPC pour multi-thread ne sont **pas disponibles** dans les statistiques collectées.

TABLE 3 – IPC par largeur du processeur et nombre de threads.

Width	1 thread	2 threads	4 threads	8 threads	16 threads
2 voies	2.128	N/A	N/A	N/A	N/A
4 voies	2.128	N/A	N/A	N/A	N/A
8 voies	2.128	N/A	N/A	N/A	N/A

— Une IPC de 2.128 indique une bonne exploitation du parallelism au niveau instruction, même avec une largeur de 2.
— Le fait que l'IPC soit identique pour 2, 4 et 8 voies suggère que l'application n'est pas limitée par la largeur de l'issue window.

Q12 : Discussion et interprétation (max. 10 lignes)

L'étude des performances du DerivO3CPU avec différents widths et nombres de threads révèle plusieurs points clés :

1) Absence d'effet du width : Les trois configurations (2, 4, 8 voies) donnent des performances **identiques**. Cela suggère que l'application (multiplication matricielle) ne peut pas exploiter plus de 2 voies d'exécution en parallèle au niveau instruction.

2) Parallelism exploité : Le speedup sub-linéaire (1.93× pour 16 threads) indique une contention sur les ressources partagées, notamment le cache L2 unifié. Les résultats montrent des taux de miss L2 très élevés (0.59 en moyenne), limitant la bande passante mémoire.

3) Scalabilité : L'architecture multicoeur montrait un bon scaling jusqu'à 8 threads, mais avec 16 threads, la contention devient significative, rendant les gains marginaux.

Configuration CMP la plus efficace

Q13 : Proposez une configuration ou une gamme de configuration de l'architecture CMP (nombre de threads de l'application test omp, nombre et type de coeurs) qui vous semble la plus appropriée si la contrainte recherchée par le concepteur du système est l'efficacité surfacique ? Discussion et interprétation (max. 10 lignes).

Q14 : Au regard de l'évolution théorique du speedup et son évolution constatée lors des questions précédentes, proposez une tentative d'explication (max. 10 lignes).