Mémoires caches - Evaluation des performances de différentes configurations de mémoires caches

1st Luiz Gariglio Dos Santos
Ingénieur Degree Programme in STIC
ENSTA Paris
Paris, France
email@ensta-paris.fr

2nd Helena Guachalla De Andrade
Ingénieur Degree Programme in STIC
ENSTA Paris
Paris, France
email@ensta-paris.fr

3rd Santiago Florido Gomez
Ingénieur Degree Programme in STIC
ENSTA Paris
Paris, France
santiago.florido@ensta-paris.fr

4th Franck Ulrich Kenfack Noumedem
Ingénieur Degree Programme in STIC
ENSTA Paris
Paris, France
email@ensta-paris.fr

***Abstract*—mamamammsmamammasaalsnoihrf eirfrf**
***Index Terms*—hsbuahbdwefwcwcbdwuguwcc**

I. Introduction

ahdubwyfwi ceuce hyec

II. Profiling de l'application

Pour procéder à l'évaluation des configurations de cache pour chacun des algorithmes proposés et analyser leurs performances, il est proposé de réaliser un *profiling* de l'application à l'aide du simulateur gem5. Le *profiling* est essentiel, car il permet de quantifier des éléments du comportement du programme afin de prendre des décisions d'optimisation et de microconception architecturale fondées sur des données, principalement issues de la simulation [1]. Il permet également d'identifier des *hotspots* sur lesquels concentrer la conception et l'optimisation, c'est-à-dire de cibler en priorité les composantes qui contribuent le plus au temps d'exécution. Enfin, il fournit une première approximation de la caractérisation de la charge de travail (*workload*) d'un programme, ce qui facilite l'orientation des choix de conception.

Classe	Dijkstra large (A7)	Dijkstra large (A15)
Lecture (Load)	45 516 963 [28.4]	45 905 506 [28.5]
Écriture (Store)	19 439 553 [12.1]	19 593 718 [12.1]
Branchement	43 904 570 [21.5]	44 122 872 [21.5]
Calcul entier (Int)	95 334 242 [59.5]	95 780 142 [59.4]
Calcul flottant (Fp)	0 [0.0]	0 [0.0]
Total d'instructions exécutées	**204 195 328**	**205 402 238**

TABLE I: Dijkstra large (Cortex-A7 vs Cortex-A15).

Classe	Dijkstra small (A7)	Dijkstra small (A15)
Lecture (Load)	10 313 882 [28.5]	10 474 419 [28.5]
Écriture (Store)	4 759 916 [13.2]	4 850 175 [13.2]
Branchement	9 823 729 [21.4]	9 978 854 [21.4]
Calcul entier (Int)	21 106 947 [58.3]	21 363 899 [58.2]
Calcul flottant (Fp)	0 [0.0]	0 [0.0]
Total d'instructions exécutées	**46 004 474**	**46 667 347**

TABLE II: Dijkstra small (Cortex-A7 vs Cortex-A15).

Classe	Blowfish (A7)	Blowfish (A15)
Lecture (Load)	19 141 [21.8]	19 769 [21.3]
Écriture (Store)	5 516 [6.3]	5 671 [6.1]
Branchement	29 760 [25.3]	30 060 [24.4]
Calcul entier (Int)	63 342 [72.0]	67 560 [72.6]
Calcul flottant (Fp)	0 [0.0]	0 [0.0]
Total d'instructions exécutées	**117 759**	**123 060**

TABLE III: Blowfish (Cortex-A7 vs Cortex-A15).

Le *profiling* montre que le calcul entier domine dans tous les cas : $\approx 60\,\%$ pour Dijkstra et $\approx 72\,\%$ pour Blowfish. Ainsi, un processeur doté de davantage d'ALU pour paralléliser ces opérations pourrait offrir un gain notable. De plus, une restructuration des boucles afin de réduire les branchements, également significatifs, pourrait améliorer les performances. Enfin, l'optimisation des accès mémoire (notamment la localité et les accès contigus) est pertinente, car les *loads* dépassent $20\,\%$. Ces constats orientent directement les choix de micro-architecture et d'optimisation logicielle.

References

[1] M. J. P. (University of York), "Profiling," *Lecture Notes (4th Year HPC), University of York*. [Online]. Available: https://www-users.york.ac.uk/~mijp1/teaching/4th_year_HPC/lecture_notes/Profiling.pdf. Accessed: Feb. 9, 2026.