

# Comparatif des performances d'une application en fonction des paramètres de compilation utilisés et analyse des performances

TER M1 Informatique

Pierre Tassel sous la direction de Sid Touati

Département Informatique  
Université Nice Sophia Antipolis

Master Informatique

18 juin 2018

# Plan

- 1 Présentation du programme
- 2 Démarche expérimentale
- 3 Analyse des performances séquentielles
- 4 Profilage du code
  - GProf
  - vTune
- 5 Difficultés de l'optimisation du code
- 6 Parallélisme de donnée
  - Approche naïve
  - False Sharing
  - Thread Affinity
  - Utilisation outils Intel
- 7 Modification Algorithmique
  - Trie coups
  - Table de transposition

# Présentation du programme

## MinMax et le jeux de l'Awale

- MinMax avec coupes Alpha Beta
- Variante du jeux de l'Awale avec 6 cases et 4 graines par case
- Ecrit en C++ par M Régin



Figure – Le jeu de l'Awalé.

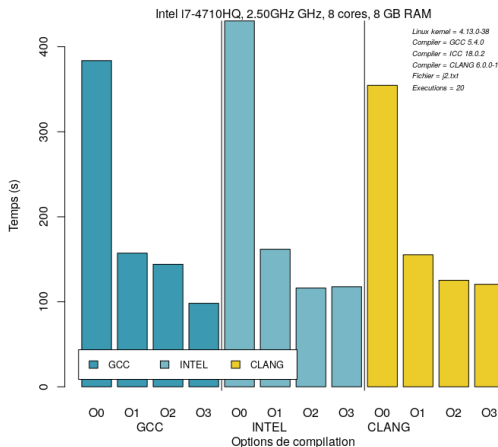
# Démarche expérimentale

## Environnement matériel et logiciel

- Processeur Scaling désactivé, processeur 8 coeurs 2,5GHz, 8Go DDR3, 250Go de SSD
- Environnement minimaliste en CLI, Arch Linux basé sur noyau 4.14.13-1
- 3 compilateurs analysés, GCC, ICC et CLang dernières versions disponible
- Fichier d'entrée généré en faisant jouer le programme contre lui même
- On répète les exécutions 20 fois pour chaque configurations testé

# Analyse des performances séquentielles

Figure – Temps d'exécution séquentiel quand le programme commence.



# Profilage du code

## GProf

% time	cumulative seconds	calls	total s/call	name
48.83	167.72	5172852992	0.00	jouer_coup(Next*, Pos*, Pos*, int, int)
18.25	230.39	3262157329	0.00	copier(Pos*, Pos*)
17.22	289.56	3262157262	0.00	est_affame(Pos*, int)
8.79	319.75	1457826958	0.00	calculer_coup(Next*, Pos*, int, int, int, int, bool)
5.31	337.98	3252245939	0.00	valeur_minimaxAB(Next*, Pos*, int, int, int, int, bool)
0.87	340.99	1775297034	0.00	evaluer(Pos*)
0.52	342.77	67	0.03	test_fin(Pos*)
0.09	343.65	33	10.34	decisionAB(Next*, Pos*, int, bool)

Figure – Analyse du code avec GProf.

# Profilage du code

vTune

## Elapsed Time <sup>?</sup>: 118.666s

➤ CPU Time <sup>?</sup> :	117.755s
Instructions Retired:	588,377,500,000
CPI Rate <sup>?</sup> :	0.499
CPU Frequency Ratio <sup>?</sup> :	1.000
Total Thread Count:	1
Paused Time <sup>?</sup> :	0s

## Top Hotspots

This section lists the most active functions in your application.

Function	Module	CPU Time <sup>?</sup>
<a href="#">jouer_coup</a>	advisor.out	70.132s
<a href="#">calculer_coup</a>	advisor.out	15.441s
<a href="#">copier</a>	advisor.out	14.165s
<a href="#">est_affame</a>	advisor.out	7.176s
<a href="#">valeur_minimaxAB</a>	advisor.out	5.755s
[Others]		5.087s

## Elapsed Time <sup>?</sup>: 120.174s

CPU Time <sup>?</sup> :	118.454s
▼ Memory Bound <sup>?</sup> :	2.7%
L1 Bound <sup>?</sup> :	6.6%
▼ DRAM Bound <sup>?</sup> :	
DRAM Bandwidth Bound <sup>?</sup> :	0.0%
LLC Miss <sup>?</sup> :	0.1%
Loads:	157,360,120,662
Stores:	100,867,825,944
LLC Miss Count <sup>?</sup> :	900,054
Average Latency (cycles) <sup>?</sup> :	8
Total Thread Count:	1
Paused Time <sup>?</sup> :	0s

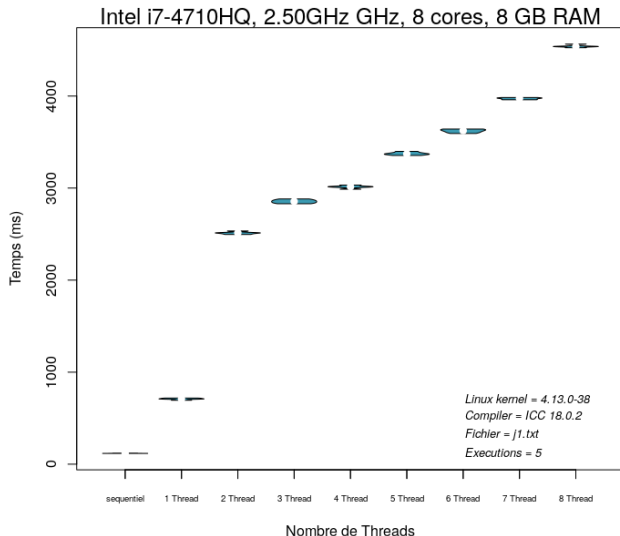
Figure – vTune analyse usage mémoire.

Figure – vTune analyse usage processeur.

- Algorithme très séquentiel
- Nombre d'itérations faibles
- Usage de pointeurs qui pourraient aveugler les compilateurs



Figure – Résultat ajout parallélisme avec compilateur Intel.



# False Sharing

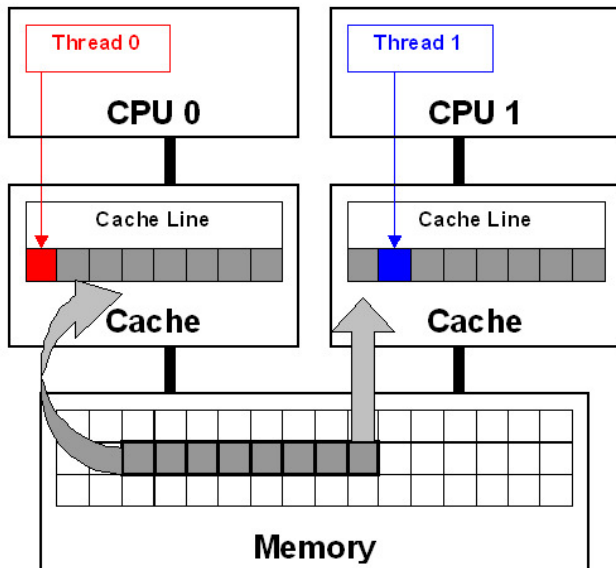
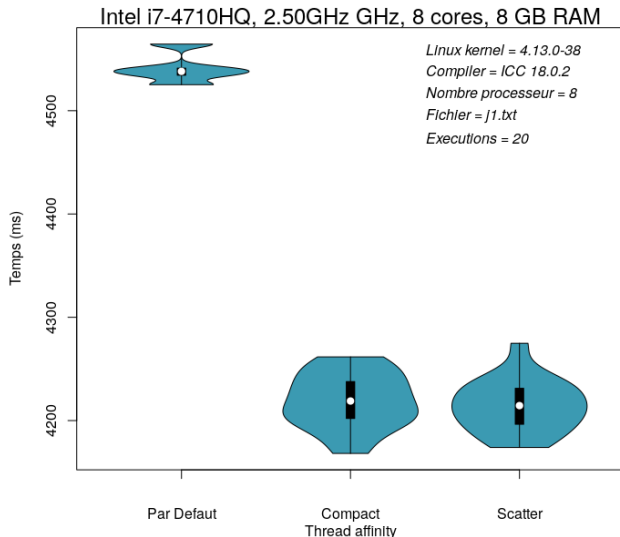


Figure – Le phénomène de False Sharing.

# Thread Affinity

Figure – Analyse de l'impact du placement des threads.



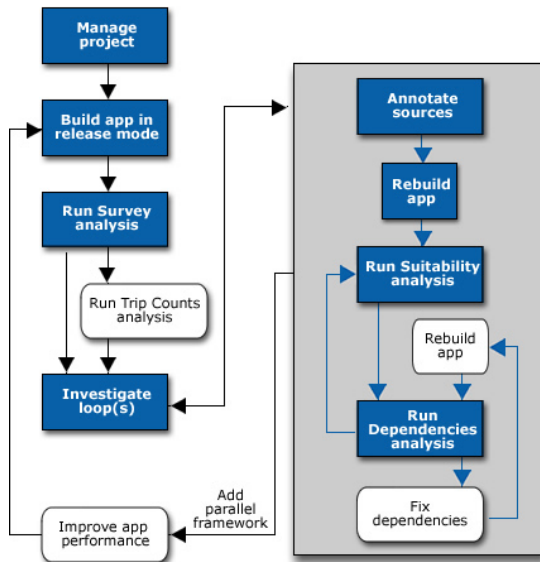
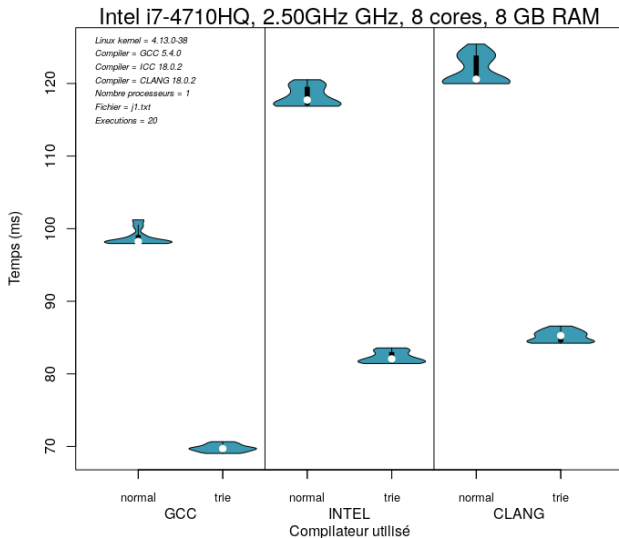


Figure – Processus itératif d'Intel pour ajout parallélisme.

- Dictionnaires ouvertures.
- Trie des coups.
- Table de transposition.

# Trie coups

Figure – Analyse de l'impact du trie des coups.



# Table de transposition

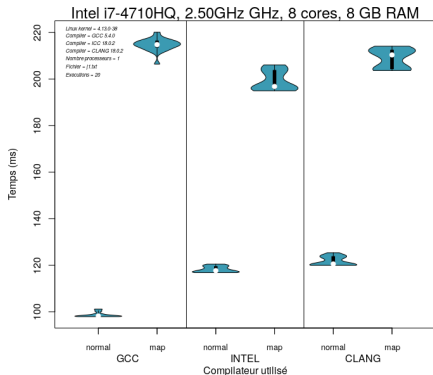


Figure – Arbre binaire de recherche.

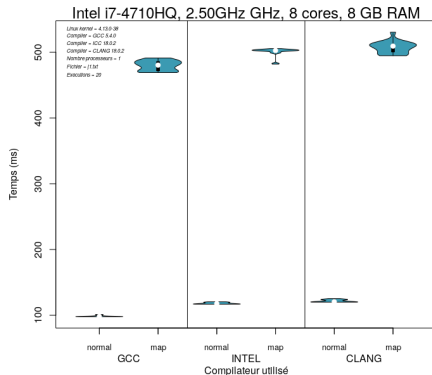


Figure – Table de hashage.