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

Pierre TASSEL

(Dated: 25 février 2018)

L'utilisation d'outils tels que Gprof et Vtune© permettent une analyse plus poussée des programmes et ouvrent la voie vers de possibles optimisations.

Dans le but de nous familiariser avec les outils d'analyse de performances d'Intel et du projet GNU, nous allons analyser un algorithme de type MinMax sur l'une des variantes du jeu Awale avec 6 cases par joueur. Le but étant de déterminer la meilleure façon de compiler ce programme (compilateur et options à utiliser) et d'analyser ce programme pour dégager des pistes vers de possibles futures optimisations.

I. DETERMINATION DE L'IMPACT DES DIVERSES MÉTHODES DE COMPILATION

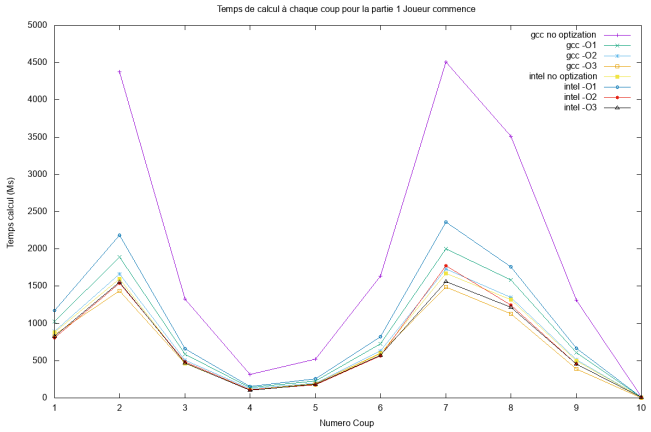
Afin d'analyser l'impact de la méthode de compilation utilisée (aussi bien le compilateur que ses options) nous allons modifier le programme afin qu'il affiche le numéro du coup joué et le temps qu'il a passé en milliseconde pour le calculer, cela nous permettra ensuite de tracer des graphes pour chaque partie montrant l'évolution du temps de calcul en fonction du compilateur et des options utilisées.

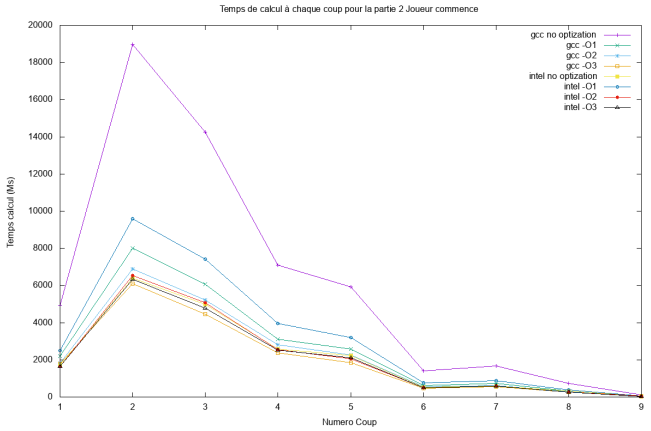
A. Méthodologie

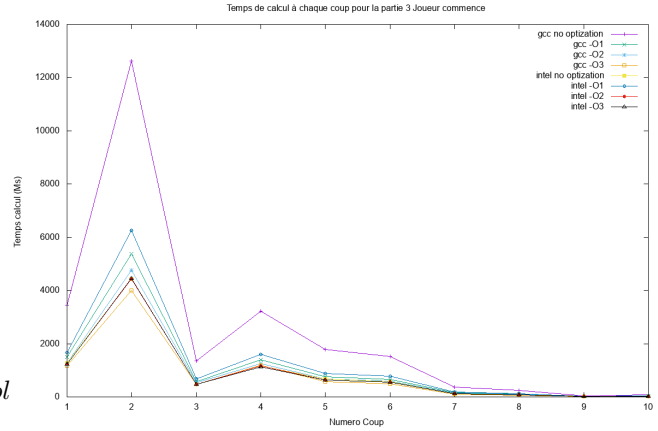
Ce programme étant déterministe (pas d'utilisation de nombres aléatoire ni de modification de la profondeur du parcours MinMax en fonction du temps) nous pouvons comparer les performances des différends programmes généré en leur faisant jouer les mêmes parties (car l'on jouera toujours les mêmes coups peu importe la configuration de compilation utilisée).
Nous ferons jouer 10 parties à chaque variante du programme (ndlr : méthode de compilation différente), 5 parties où nous commençons à jouer et 5 parties où le programme commence à jouer. Ensuite, nous comparons le temps pris pour calculer chaque coup.
Nous avons créé des fichiers texte représentant les entrées relatives à la partie (disponible dans le dossier "input") et nous avons rediriger l'entrée standard vers ces fichiers. Puis nous avons modifier la sortie pour qu'elle sorte des données sous la forme $numero_coup$: $temp_ecoulee, afindepouvoirtracerdesgraphesavecl'outilGnupl$

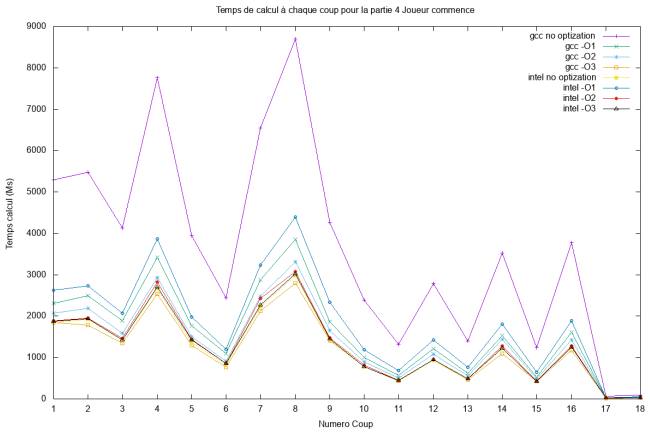
B. Résultats

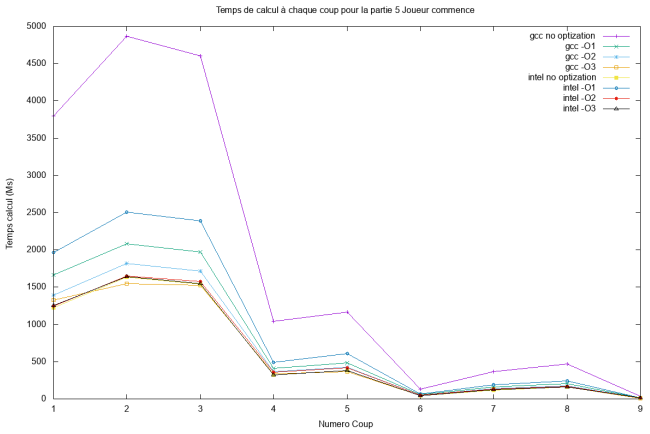
1. *Le joueur commence*





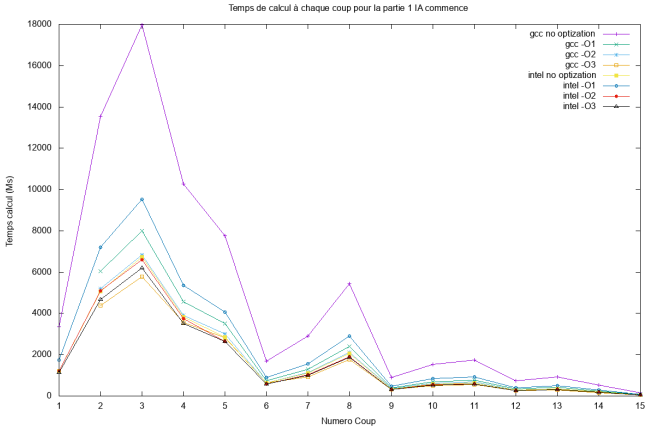


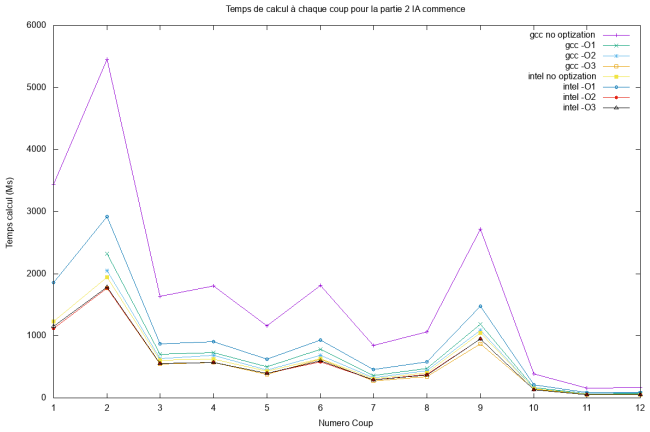


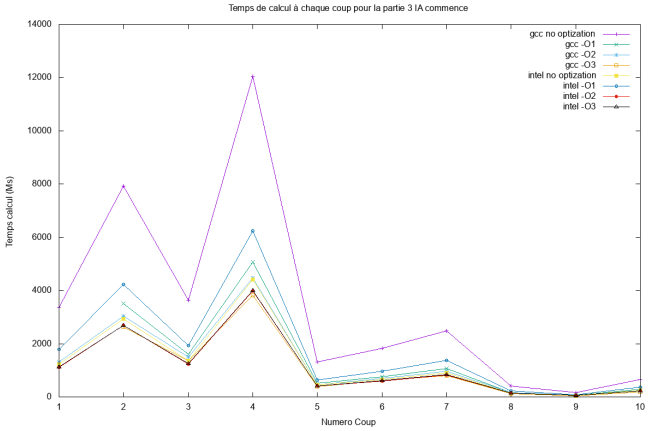


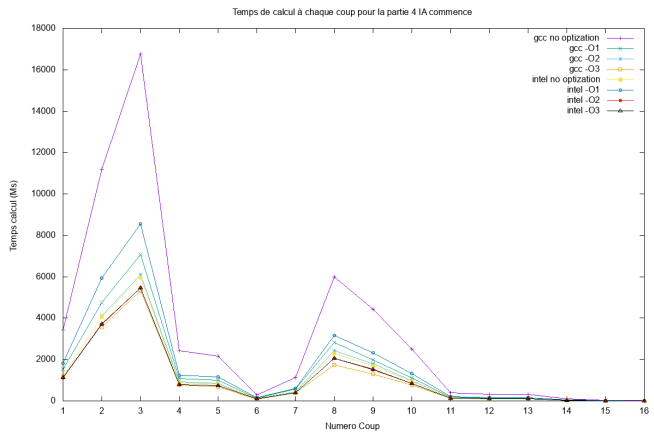
On remarque que la compilation avec GCC (g++) accompagné de l'option -O3 est la plus rapide quasiment ex eaquo avec le compilateur Intel (icl) accompagné lui aussi de l'option -O3, ce résultat est étonnant, car on pourrait s'attendre à ce que le compilateur Intel fasse mieux que celui de GCC sur un processeur Intel.
Les résultats avec GCC sont plus disparates en fonction de l'option utilisé. De base (sans options d'optimisation) l'exécutable produit est beaucoup plus lent que les autres exécutables produit par d'autres méthodes de compilation. Au contraire de base l'exécutable produit par le compilateur d'Intel est proche de l'optimal.
On remarque que la compilation avec l'option -O1 est plus lent que sans option de compilation et le temps d'exécution avec -O2 et sans option d'optimisation est quasiment similaire, ce qui laisse à penser que de base le compilateur d'Intel compile avec -O2 comme option activée par défaut.

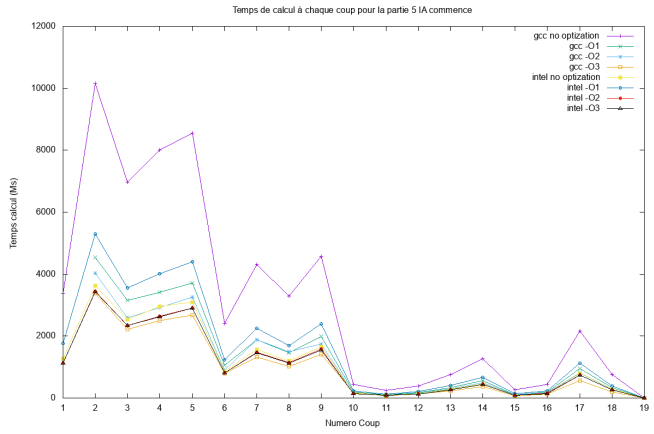
2. L'IA commence





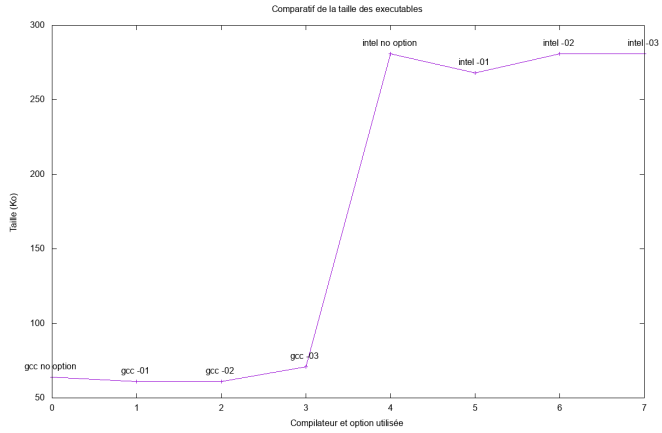






Sans surprise, on observe des résultats similaire quand c'est l'IA qui commence.

3. *Taille de l'éxécutable*



Le compilateur d'Intel quel que soit l'option produit des exécutables beaucoup plus lourd (en taille) que le compilateur GCC. Les exécutables produit par le compilateur d'Intel sont quasiment 4 fois plus lourd que ceux de GCC.

C. Conclusions

Pour ce programme la meilleure option de compilation est l'usage de GCC avec l'option -O3.

II. UTILISATION D'OUTILS D'ANALYSE DE PERFORMANCE DES PROGRAMMES

Jusqu'à présent nous avons exécuté le programme directement afin de tester quelle méthode de compilation était la plus approprié. Pour obtenir des informations un peu plus poussé nous allons utiliser les logiciels VTune et Gprof.

A. Gprof

Gprof nous apprend que la méthode qui prend le plus de temps de calcul CPU est la méthode calculer_coup, ainsi que cette méthode est le plus souvent appelé depuis la méthode main.

B. VTune

Pour l'occasion le programme à été recompilé avec les options de compilations recommandé par Intel : `https://software.intel.com/en-us/vtune-amplifier-help-compiler-switches-for-performance-a`
VTune est un peu plus complet et permet d'analyser non seulement l'exécutable produit par Icl (ndlr : le compilateur d'Intel) mais aussi GCC.
Etonnement VTune semble indiquer que c'est la fonction jouer_coup qui prend le plus de temps CPU. Ce résultat reste compréhensible sachant que la fonction calculer_coup appel la fonction jouer_coup. On y apprend aussi que le parallélisme est inexistant, le programme n'utilise qu'un seul coeur de la machine à la fois.

C. Conclusion

D'après les données recueillis au travers de ces deux logiciels, on peut en déduire qu'il faut concentrer les efforts d'optimisation sur la fonction calculer_cout et la fonction jouer_coup.
Essayer d'introduire du parallélisme semble crucial à l'optimisation des performances.

III. CONCLUSION

Au travers de ces deux analyses nous avons réussi à déduire la façon optimale de compiler ce programme et les pistes à envisager pour optimiser les performances de cette application.