Documentation de la performance énergétique

G7, gl35

January 2023

Contents

1 Introduction

A lui seul, le numérique représente environ 4% des émissions de gaz à effet de serre dans le monde. Et ce chiffre pourrait doubler d'ici à 2025, ordinateurs, smartphones, objets connectés et autres data centers se montrant particulièrement gourmands en électricité. Etant des futurs ingénieurs responsables et étudiants dans l'une des grandes écoles de numériques responsables, nous nous soucions de l'efficience énergétique de notre produit. Les deux critères d'évaluation considérés sont l'efficience du code produit et le processus de validation.

2 Code assembleur généré

Afin d'optimiser le code assembleur généré, une première idée a été d'essayer de générer moins d'instructions assembleur, en utilisant les différents mode d'adressage et la méthode **dval** défini. Nous aurions pu s'inspirer de cette optimisation afin de fournir un code encore plus performant mais cela n'a pas été possible lors du temps imparti. D'autres idées d'amélioration sont de remplacer les multiplications par des décalages de bits lorsque c'est possible, ou encore de calculer directement les expressions constantes à la compilation. Malheureusement, ces optimisations n'ont pu être implémenté dans la limite du temps imposé.

3 Processus de validation

L'étape de validation est l'étape la plus gourmande au niveau énergétique. En effet l'exécution de tous les tests (unitaires, décompilation, analyse lexicale, contextuelle, génération de code, ...) dure presque à l'aide de la commande `mvn test`.
Durant le projet l'utilisation de la commande précédente était rare à voir nulle. En effet, chaque membre lance les tests voulus à la main (cela prend même pas une seconde pour chaque test au lieu de plusieurs minutes) car il 'aurait besoin que d'un test contenant une grande partie des fonctionnalités qui intéresse le développeur concerné ce qui est moins énergivore et beaucoup plus rapide.
En plus la majorité des tests lancés étaient sur un ordinateur portable consommant 3 fois moins d'énergie qu'un ordinateur de bureau.
Un ordinateur de bureau consomme en moyenne entre 75 et 730 kWh d'électricité en fonction de son utilisation, et coûte entre 15 et 130 € par an.La consommation d'électricité d'un ordinateur portable est en général comprise entre 20 et 275 kWh par an, pour un prix de 4 à 50 €.,
Source : hellowatt.fr.

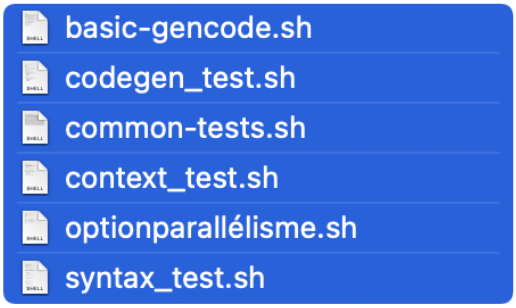

Figure 1 : Scripts Ajoutées

De la même façon le lancement de l'outil jacoco était limité, et seulement par un membre de l'équipe qui communique les résultats aux autres membres.

4 Extension

Les instructions de notre code ne sont qu'un ensemble d'instructions assembleur qui lors de leur exécution consomment de l'énergie, faible certes, mais qui devient non négligeable si cumulées.

Ce qui veut dire que réduire le nombre d'opérations d'un algorithme permet de réduire la consommation en terme d'énergie.

Notre but était alors de chercher des algorithmes qui nous permettent d'implémenter nos fonctions trigonométriques avec un bon compromis entre précision de calcul et nombre d'opérations effectuées. On a alors testé plusieurs algorithmes au départ, comme celui de Cordic qui nous semblait bon au niveau de la précision, mais qu'on n'a pas choisi au final vu qu'il nécessite beaucoup de calculs, et par conséquent consomme plus d'énergie. En effet, il lui faut une implémentation d'une table de valeurs au départ.

On a alors retenu la méthode des séries de Taylor pour les fonctions sin, Cos et Asin. Et on a choisi les polynômes d'Hermites pour la fonction Atan. On ne s'est pas arrêté là. Lors de nos implémentations, on a essayé d'optimiser nos algorithmes tout en réduisant les calculs. Pour les fonctions Sin et Cos, au lieu d'utiliser un polynôme d'ordre 15 qui nous donnait de bons résultats, mais qui occupait un peu de la mémoire, on a privilégié un polynôme d'ordre 8 pour le cos(9 pour le sin). On a aussi dû réduire l'intervalle de départ pour améliorer la précision et diminuer les calculs, en effet, on est passé de $[-2\pi, 2\pi]$ à $[0; \pi/4]$, ce qui divisent le nombre de calculs par un facteur de 8 et par conséquent la consommation énergétique. En ce qui concerne l'Atan, on a écrit les coefficients du polynôme utilisé en dur, afin d'éviter de les recalculer à chaque fois, sachant qu'ils nécessitent de lourds calculs comme des factoriels, des divisions(40 cycles d'horloge) et des multiplications(20 cycles d'horloge).

Quelques résultats trouvés pour nos fonctions, sachant que l'algorithme de Cordic nécessite au minimum du 100 000 cycles d'horloge :

fonction	nombre de cycles
Sin	9000
Cos	9000 à 28000
Asin	80000
Atan	10000

Ces résultats peuvent être justifiés comme suit :
Pour les fonctions sin et cos, on calcule à chaque fois les coefficients de leurs séries de Taylor, car lorsqu'on a essayé de les écrire en dur, on n'avait pas une bonne précision au-delà de pie. On remarque aussi que le nombre de cycles d'Asin est élevé par rapport aux autres. Cela est dû au degré élevé du polynôme de série de Taylor utilisé dans son implémentation. Et finalement,

pour la fonction Atan, on a laissé des opérations de divisions qu'on pouvait mettre en dur, et économiser alors 40 cycles d'horloge pour chaque division.

5 Palmarès du Projet GL 2023

Les scores obtenus sur les fichiers de performances fournies en utilisant la commande no check `decac -n` sur le fichier .deca et `ima -s` sur le fichier .ass sont :

- **ln2.deca** : Nombre d'instructions: 94 ,temps d'exécution : 15482 (L0) .
- **ln2_fct.deca** : Nombre d'instructions: 141, temps d'execution: 18793(L1).
- **syracure42.deca** : Nombre d'instructions: 38, temps d'exécution: 1306 (S).

Au final, nous obtenons un score total de : (L0 + L1 + 10*S) = **47335**.