Grenoble INP - Ensimag

Documentation sur l'impact énergitique

PROJET-GL GROUPE 49

BENTAIBI Oussama
AÏT HAMMOU Driss
AJJA Hamza
DAOUD YOUSSEF
LEACHOURI Khalil

2^{nd} Year - 29 janvier 2021

I Introduction

Dans notre monde moderne, la dégradation de l'environnement est l'un des problèmes majeurs qui acculent notre planète en ce moment. Et par conséquent les scientifiques visent , par chaque produit qu'ils créent, à diminuer son impact négatif sur l'environnement. D'où parmi les buts du projet GL, considéré comme projet formateur, on trouve la formation d'un ingénieur qui respecte son environnement en essayant de réduire le maximum de pertes énergétiques possible. Afin d'atteindre ce but, notre groupe a essayé de diminuer les effets énergétiques de notre compilateur en utilisant les stratégies suivantes :

II Exécution en parallèle

Pour diminuer le temps de compilation des fichiers .deca, on a eu recours au MultiThreading. En effet, un thread est une unité d'exécution faisant partie d'un programme. Cette unité fonctionne de façon autonome et parallèlement à d'autres threads. Le principal avantage des threads est de pouvoir répartir différents traitements d'un même programme en plusieurs unités distinctes pour permettre leurs exécutions simultanées. En plus, l'exécution en parallèle améliore généralement les performances et donne les résultats rapidement par rapport à l'exécution ordinaire, surtout si la machine possède plusieurs cœurs. Dans notre implémentation, on a bien insisté sur l'implémentation de l'option "-P" vu son impact énergétique sur notre compilateur. En effet, on remarque dans la figure ci-dessous la différence très claire entre la compilation ordinaire et la compilation en parallèle.

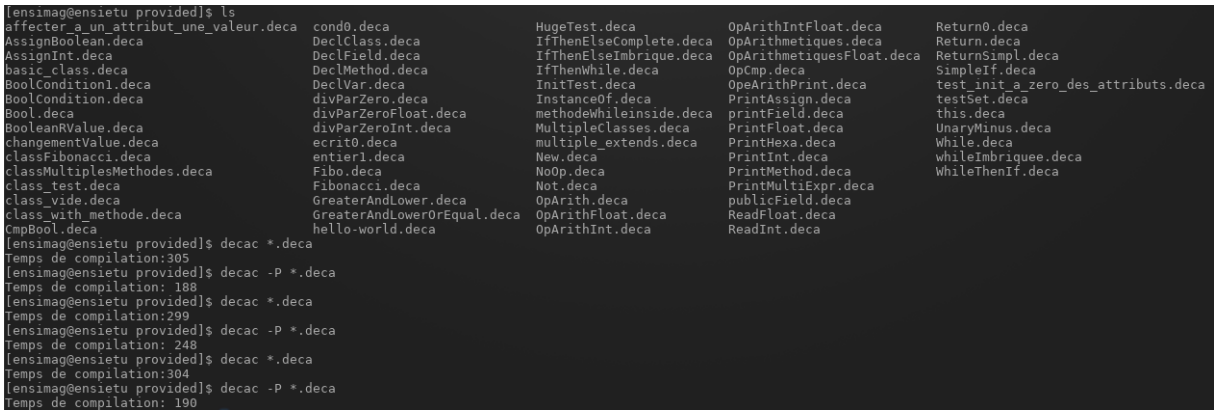

FIGURE 1 – exemple de tests avec et sans l'option -P

Ainsi, on peut dire que les 71 fichiers .deca qui se trouvent dans le répertoire (/src/test/deca/codegen/valid/provided) se compilent dans un temps moyen de 302.67 ms sans utiliser l'option "-P". Cependant, avec cette option le temps moyen diminue jusqu'à 208.67 ms

III Impact Énergétique de l'extension

Pour la partie de l'extension, on cherche à avoir une bonne précision dans un temps minimal. L'impact énergétique était un des facteurs de choix des algorithmes :

III.1 Cordic / Taylor

Dans ce cas, on constate que l'algorithme de Cordic prend vraiment beaucoup de temps par rapport à celui de Taylor vu que le premier utilise plus d'opérations. Par exemple, en testant la fonction sinus sur 205888 valeurs entre 0 et 2Pi, on constate que l'algorithme de Cordic prend 209384 ms tandis que celui de Taylor prend 1278 ms.

III.2 Taylor avec Horner / Taylor sans Horner

Pour bien améliorer la partie énergétique de notre extension on a pensé à faire de petites modifications sur le calcul de cosinus et sinus. En effet, on a préféré utiliser les polynômes de Horner au lieu de faire un calcul classique des séries de Taylor. Les polynômes de Horner nous ont aidé à diminuer le nombre d'opérations effectuées. En fait, on a passé de n additions et 2n - 1 multiplications à n additions et n multiplications. En plus, on a bien remarqué la différence dans le temps lors des tests. On a pu passer de 5030 ms à 3060 ms dans l'exécution de la fonction cos sur 1647100 valeurs entre 0 et 2π.

IV Conclusion

En guise de conclusion, on peut dire qu'on a économisé une grande quantité d'énergie à travers le MultiThreading. Et concernant les algorithmes de l'extension, on a essayé de faire une sorte d'équilibre entre la précision et le temps d'exécution.