PHELMA GRENOBLE INP
ENSIMAG GRENOBLE INP

Impact énergétique

DIJS Thomas FALGAYRAC Loic HO-SUN Jules
NOIRY Sylvain WANG Caroline
groupe GL 54

Phelma SEOC/ Ensimag ISI-MMIS
28 janvier 2022

Table des matières

1 Introduction

Même s'il est évident que notre compilateur n'aura pas un impact environnemental démesuré, il est primordial de s'en préoccuper dès à présent afin de prévoir l'impact de nos futurs travaux, qui sera sûrement plus important.

L'informatique représente en effet une grande part des émissions de gaz à effets de serre, entre 2 et 10%, mais est aussi à l'origine d'une importante pollution des sols, notamment due aux terres rares. Limiter cet impact n'est pas seulement une envie mais surtout une nécessité.

2 Consommation énergétique des machines

La mesure de l'impact énergétique d'un projet de développement logiciel s'effectue en très grande majorité en terme de consommation énergétique. Nous allons détailler ci-dessous comment nous avons mesuré la consommation énergétique de nos machines.

2.1 Nos machines personnelles

Pour évaluer la consommation énergétique des machines, il faut tout d'abord les données constructeur des machines sur lesquelles le projet est réalisé. Nous avons tous réalisé le projet sur nos machines personnelles, des ordinateurs portables.

Une batterie d'un ordinateur portable fait environ 42 Wh. Les membres du groupe rechargent leur ordinateur portable en moyenne deux fois par jour. On peut supposer que la batterie tient deux fois le temps de la recharge, donc il y a besoin de trois fois la capacité totale de la batterie par jour et par ordinateur. Il y a 5 ordinateurs sur 4 semaines ; comptons aussi un jour par week-end. Cela nécessite donc au total 15 kWh, ce qui représente 2 jours de chauffage dans un logement moyen. Le pur coût énergétique de ce projet n'est donc pas trop important. L'utilisation d'ordinateurs portables est enfin moins énergivore que celle d'ordinateurs fixes (environ 33% moins énergivores).

Il ne faut pas oublier les coûts cachés. Une recherche internet demande en moyenne 0.0003 kWh (données Google). Nous faisons en moyenne 50 recherches internet par jour et par membre du groupe. Cela représente sur la durée totale du projet 1,8 kWh ; c'est bien moins important que la consommation de nos machines personnelles.

Néanmoins, la part la plus importante de la consommation énergétique d'un ordinateur reste sa fabrication, qui nécessite environ 900 kWh d'énergie et 50 kWh pour le transport. Comme les ordinateurs sont fabriqués en Chine, qui produit son énergie principalement à base de charbon, la fabrication d'un ordinateur émet près d'une tonne de CO2. Il ne faut cependant pas imputer la totalité de ce coût au projet, car nos machines personnelles n'ont pas été achetées exclusivement pour celui-ci. Néanmoins la compilation du projet étant assez gourmande, un membre de l'équipe a précipité le changement de son ordinateur pour ne trop attendre.

2.2 Impact des choix de compilation de deca vers ima

L'environnement de travail ima utilise la même architecture que le processeur Motorola 68k, sorti en 1979. Ce processeur est assez vieux mais son architecture reste d'actualité, puisque de nombreux systèmes embarqués se contentent de microcontrôleurs avec des performances similaires, mais beaucoup moins chers et suffisants pour les tâches souhaitées par rapport à leurs homologues pour PC. Choisir des processeurs utilisant des technologies matures permet d'utiliser des usines déjà construites et donc de diminuer le coût énergétique et de construction.

3 Efficience du code produit

Un code performant augmente la durée d'utilisation du produit, puisque le besoin d'avoir un produit plus performant ne se fera ressentir que plus tard.

Il est possible de mesurer la performance du code grâce à l'efficience du code produit : cette efficience est caractérisée par la facilité d'exécution du code un fois converti en assembleur pour les machines.

3.1 Mesure en terme de temps

Dans le répertoire */usr/bin/time* se trouve la commande `time`. Cette commande s'exécute dans n'importe quel répertoire en tapant la commande suivante :

`time <nom_du_programme>`

Cette commande retourne trois données :

— *real* qui représente le **temps réel** d'exécution du fichier.
— *user* qui représente le **temps CPU réel** utilisé pour exécuter le processus.
— *sys* qui représente le **temps passé dans le noyau** au sein du processus.

Ainsi, le temps total passé par le processeur est donné par l'addition $user + sys$. Ce temps est très souvent supérieur au temps total d'exécution puisque, de nos jours, les processeurs ont de nombreux cœurs.

Néanmoins, cette commande est assez inutile lorsqu'il s'agit de mesurer uns consommation énergétique. En effet, des ordinateurs fixes très puissants prendront très peu de temps à exécuter des instructions, alors que des ordinateurs beaucoup plus vieux ayant une consommation moindre en nécessiteront beaucoup plus. Il faudra donc corréler ce temps d'exécution avec la quantité d'énergie utilisée par l'ordinateur.

3.2 Mesure en terme de cycles d'horloge

La manière la plus simple pour mesurer l'efficacité d'un code serait l'utilisation de la notion de temps d'exécution. Néanmoins, cette méthode diffère suivant le nombre de processus utilisés pour des tâches de fond sur l'ordinateur mais aussi et surtout suivant la puissance des ordinateurs en question.

Nous allons donc utiliser dans cette question un autre moyen de mesure de l'efficacité du code : le comptage du nombre de cycles d'horloge nécessaires à l'ordinateur pour l'exécuter. Pour exécuter le test de performances Syracuse, il nous faut par exemple 735 cycles d'horloges. Pour un processeur de 1.6 GHz, il faut donc :

$$T_{excution} = \frac{735}{1.60 * 10^9} \approx 4.59^{-7} s$$

A l'échelle humaine, ce temps d'exécution n'est pas perceptible. Pour un processeur nécessitant 100W de puissance lorsque ses capacités sont utilisées au maximum, il aura consommé $4.59^{-5} J$, soit une quantité d'énergie équivalente à celle dégagée par des collisions de noyaux d'ions lourds. C'est donc très faible en terme de calculs purs.

3.3 Implémentation

Nous avons pu obtenir ce résultat très faible en se concentrant sur l'optimisation de la génération de code. L'ensemble des optimisations réalisées

nécessitent un temps de compilation plus important mais permettent de limiter grandement le temps d'exécution. Parmi les optimisations réalisées, on retrouve l'utilisation d'un graphe, l'élimination du code mort ou encore la propagation des constantes *(voir documentation de l'extension)*.

Pour ne pas nuire à la qualité du compilateur, dans un premier temps, nous n'avions pas pris en compte l'aspect énergétique. Au fur et à mesure de la diminution du nombre de cycles nécessaires, nous avons comparé les différences des résultats, et lorsqu'ils n'étaient pas identiques, nous recommencions. En effet, il valait mieux que le compilateur fonctionne bien quitte à nécessiter quelques cycles d'horloge de plus plutôt qu'il engendre des erreurs : l'impact énergétique ne doit pas primer sur la qualité.

Le compilateur que nous avons fait fonctionne plutôt bien et occupe même la première place au palmarès pour le nombre de cycles nécessaires après compilation.

4 Efficience du procédé de fabrication

Le procédé qui demande le plus de temps à l'ordinateur reste la compilation du compilateur. L'opération *mvn compile* nécessite entre 3 et 50 secondes suivant les ordinateurs pour s'exécuter, soit jusqu'à $3200J$ d'énergie seulement pour le processeur. Compiler le compilateur est donc coûteux en énergie, il faut ainsi le compiler le moins souvent possible. La commande *mvn verify* prend quant à elle jusqu'à une minute.

4.1 Développement

Pendant la durée du développement, nous nous sommes efforcés de nuire le moins possible à l'environnement. Ainsi, tous les membres du groupe viennent soit en vélo, soit en tramway, deux moyens de transports parmi les plus propres, alors que les transports représentent en France 31% des émissions de gaz à effets de serre.

Nous avons regretté de ne pas pouvoir baisser le chauffage dans les salles de l'Ensimag car nous avons trouvé à l'unanimité qu'elles étaient parfois trop chauffées.

4.2 Des efforts à poursuivre

Malgré tout, les efforts restent à poursuivre. Parmi les voies d'amélioration, on aurait pu essayer de travailler 1 ou 2 jours par semaine en télétravail. Le développement n'émet pas de déchets en particulier car ne produit pas de matériel à proprement parler, il est donc assez difficile de trouver d'autres voies d'amélioration.

Ce n'est pas le cas pour nous, mais dans une entreprise qui doit vendre un produit, effectuer un maximum de tests pour s'assurer que le logiciel fonctionne parfaitement permet d'éviter les bugs, de faire des mises à jours, des échanges de datas *évitables*, et par voie de fait, de limiter la consommation énergétique du projet et les émissions de gaz à effet de serre.

5 Conclusion

Ainsi, s'attarder sur l'impact énergétique de notre projet nous aura permis de nous rendre compte que tout projet, aussi minime soit-il, a un impact sur l'environnement. Il est un devoir éthique de limiter cet impact et de faire en sorte que nos futurs projets, en laboratoire ou en entreprise, respectent la plus possible cette dimension environnementale.

En programmation, il est d'autant plus *facile* de le faire étant donné que la consommation d'un programme informatique se limite aux cycles d'horloges nécessaires pour l'exécuter. Il ne faut cependant pas oublier les coûts énergétiques cachés comme ceux nécessaires à l'exploitation des ressources premières et à la fabrication des ordinateurs, qui pour la plupart, deviennent obsolètes au bout d'une décennie.