Numérique responsable : mesurer l'évolution de l'empreinte mémoire d'un logiciel suivant ses versions en utilisant le gestionnaire de paquet Nix

Mots-clefs : Durabilité, Sobriété numérique, Gestionnaire de paquet Nix.
Niveau d'études : Licence ou M1.

Contexte

L'informatique évolue depuis des décennies en suivant une croissance exponentielle (en particulier la loi de Moore qui affirme que le nombre de transistors sur une puce double tous les deux ans, qui s'est assez bien vérifiée en pratique depuis les années 70). Cette croissance atteint ses limites : la consommation d'énergie liée au numérique est aujourd'hui importante (2 à 4% des émissions mondiales de gaz à effets de serre) et également en augmentation exponentielle (doublement tous les dix ans environ) et les ressources en termes de minerais rares sont limitées et ne pourront suivre une croissance exponentielle à long terme. Les experts du changement climatique (comme le GIEC) proposent au contraire des scénarios où les émissions de gaz à effets de serre décroissent pour atteindre la neutralité carbone autour de 2050.

S'il est tentant de penser que les progrès technologiques suffiront à réduire la consommation d'énergie et l'utilisation de ressources, l'histoire jusqu'ici montre plutôt le contraire : les progrès en efficacité énergétique ont presque toujours été compensés par une augmentation des usages (par exemple, les téléphones mobiles d'il y a 20 ans étaient conçus sur des technologies bien moins efficaces énergétiquement que celles des smartphones, et pourtant avaient une autonomie sur batterie beaucoup plus longue). On parle pour décrire ce phénomène d'« effet rebond ».

Il est donc important de penser à une réduction des usages pour permettre de réels progrès sur l'impact écologique du numérique : envisager un futur où la quantité de calculs réalisés sur l'ensemble de la planète décroît d'année en année au lieu de croître indéfiniment. Il est très difficile de prédire l'évolution du numérique sur les décennies à venir, mais cette décroissance risque d'être imposée à l'humanité faute de .

Beaucoup de chercheurs pensent que ce changement de paradigme doit être préparé et anticipé, et qu'il est urgent de réfléchir à un numérique souhaitable et durable sur le long terme. C'est la motivation de la création de l'équipe Phénix, au laboratoire CITI à la Doua.

Les outils permettant d'exécuter du code sur des architectures minimalistes (microcontrôleur, ordinateurs avec très peu de mémoire, etc.) ont déjà un très bon niveau de maturité. La manière classique de programmer ces architectures est d'utiliser un ordinateur puissant pour le développement et la compilation croisée vers une architecture peu puissante (c'est-à-dire, utiliser un compilateur générant du code pour l'architecture minimaliste, mais qui s'exécute sur un PC classique). Dans une optique de décroissance globale, il serait au contraire souhaitable, et probablement nécessaire, que l'écosystème complet permettant le développement et l'exécution puisse tourner sur une architecture minimaliste. Une chaîne d'outils capable de se compiler elle-même est dite *méta-circulaire*.

Partant de l'hypothèse que la consommation mémoire est le premier point bloquant pour exécuter un logiciel sur un ordinateur peu puissant, nous avons développé un outil de mesure de consommation mémoire et un ensemble de scripts permettant de lancer

cet outil sur divers programmes : des benchmarks synthétiques de quelques dizaines de lignes de code, et de gros programmes comme GCC, LLVM, etc.

Nous souhaitons maintenant étudier l'évolution de la consommation mémoire d'un même programme (par exemple, GCC) au fil de ses versions, idéalement depuis la version initiale (1987 dans le cas de GCC). Pour cela, il est nécessaire d'automatiser la compilation du logiciel pour un grand nombre de versions du logiciel (et de ses dépendances).

Un gestionnaire de paquet comme Nix (`https://nixos.org/`) semble approprié pour réaliser cette automatisation : il permet un contrôle très fin sur les dépendances (assurant en particulier qu'on ne dépend jamais d'une dépendance non-déclarée), une très bonne reproductibilité (évitant l'écueil « ça marche sur ma machine, je ne comprends pas pourquoi ça ne marche pas sur la tienne »). Nix permet d'installer facilement plusieurs versions du même logiciel en parallèle sur la même machine. Une autre solution similaire serait GNU Guix (`https://guix.gnu.org/`), un autre gestionnaire de paquet basé sur les mêmes principes.

1 Objectif du stage

L'objectif du stage est d'étudier plus précisément l'applicabilité de Nix pour notre cas, et le mettre en œuvre dans le cas d'au moins un logiciel à étudier. Ceci inclue :

- Prise en main du système Nix
- Prise en main de l'outil mprobe, développé dans le cadre du même projet que ce stage
- Mise en place d'un environnement de travail basé sur Nix permettant de compiler et d'installer une vieille version d'un logiciel
- Instrumentation du système de compilation et installation pour permettre de mesurer la consommation mémoire de cette version (si on prend l'exemple d'un compilateur comme GCC, cela inclue la consommation mémoire pour compiler un programme donné, et la consommation mémoire pour se compiler lui-même, le *bootstrap*).
- Automatisation du processus pour appliquer cette mesure à plusieurs versions (si possible, toutes les versions)
- Visualisation des données produites. Au minimum nous voudrions pouvoir tracer la courbe de la consommation mémoire (pic de consommation en mémoire résidente en RAM) au fil des années.

2 Comment candidater ?

Envoyer un mail à `matthieu.moy@univ-lyon1.fr`, `guillaume.salagnac@insa-lyon.fr` avec CV, quelques lignes de motivation, et éventuellement des questions. Vous pouvez joindre tout document que vous jugez utile (rapport de stage précédent, etc.).

Encadrement

- Matthieu Moy, maître de conférences UCBL/LIP `https://matthieu-moy.fr/`
- Guillaume Salagnac, maître de conférences INSA/CITI `https://perso.citi.insa-lyon.fr/gsalagnac/`