Millian Poquet
Maître de conférences
Univ. Toulouse, IRIT, équipe Sepia
IRIT2-469, 120 cours Rose Dieng-Kuntz, 31400 Toulouse
millian.poquet@utoulouse.fr

Université de Toulouse

Modélisation et simulation de l'interférence d'applications CPU monothread sur une machine

Contexte

Il est courant que des applications différentes partagent les mêmes ressources physiques. C'est en particulier le cas des services de type *Cloud* qui sont le plus souvent exécutés dans des environnements virtualisés (machines virtuelles, conteneurs…) dans des centres de calcul dédiés, couramment appelés *data centers*. La *bonne* gestion de ces infrastructures de calcul est importante pour de multiples raisons. On peut par exemple souhaiter réduire l'impact carbone du numérique ; cette empreinte est forte, en hausse, et les *data centers* sont une partie importante de cette empreinte[1].

Faire une évaluation scientifique de différentes politiques de gestion de ressources sur ces infrastructures nécessite différentes variations des entrées et des paramètres des algorithmes pour avoir du sens, ce qui conduit irrémédiablement à des besoins en temps et en espace pharaoniques. La plupart des expérimentations sont donc faites en simulation, qui rend ces études possibles en allant plusieurs ordres de grandeurs plus rapidement que des exécutions réelles et en demandant très peu de ressources.

Malheureusement, peu de simulateurs ont vu leur comportement validé contre des exécutions réelles, ce qui nuit à la confiance qu'on peut accorder à ces études. En particulier, la plupart des simulateurs ignorent complètement les interférences entre applications s'exécutant sur la même machine, ou ne considèrent ces interférences que de manière très *naïve*, alors qu'on observe en vrai que certaines applications y sont très sensibles alors que d'autres le sont beaucoup moins.

Objectifs du projet

Dans ce projet, des données vous seront fournies sur l'exécution de différentes applications extrêmement simples (*e.g.*, une boucle de `nop`, des calculs stables en mémoire n'utilisant qu'une ligne de cache, des calculs stables en mémoire utilisant un peu plus de cache, des calculs faisant le même appel système en boucle…). Vous aurez accès au code source de chaque application, à la manière dont elle est compilée, au binaire généré par le compilateur, et au contexte d'exécution de l'application. Ce contexte d'application permet de savoir comment chaque application a été lancée côté *paramètres système* : limites d'utilisation du CPU, limites sur les cœurs utilisables, lancement natif, dans un conteneur ou une machine virtuelle… Les applications peuvent être lancées seules ou non. Vous saurez pour chaque cas quel ensemble d'applications a été lancé et le progrès au cours du temps de chaque application. Vous saurez sur quelle machine l'application s'est exécutée ; différents types de machines étant ici utilisés. Si les données fournies ne sont pas suffisantes, vous pourrez exécuter d'autres applications sur d'autres machines de manière contrôlée sur la plateforme d'expérimentation Grid'5000.

Le but de ce projet est de modéliser les machines et les applications dans le but de prédire comment l'exécution des applications devrait se dérouler au cours du temps. L'idée principale est de se baser sur vos connaissances en architecture matérielle et en couches logicielles basses pour identifier quels sont les *bottlenecks* matériels et logiciels sont probables à l'exécution d'une application quand elle s'exécute seule sur une machine, ou quand plusieurs applications s'exécutent en même temps sur une machine. Par exemple il est probable d'observer qu'une application qui parcourt

[1] https://ecoresponsable.numerique.gouv.fr/actualites/actualisation-ademe-impact/

Millian Poquet
Maître de conférences
Univ. Toulouse, IRIT, équipe Sepia
IRIT2-469, 120 cours Rose Dieng-Kuntz, 31400 Toulouse
millian.poquet@utoulouse.fr

énormément de mémoire et qui fait peu de calculs soit le plus souvent limitée par la bande passante atteignable entre la RAM et le CPU, alors qu'on s'attend à ce qu'une application qui ne fait que des `nop` soit très peu sensible à cette bande passante.

On ne souhaite pas ici *juste* catégoriser les applications par leur *bottleneck* le plus probable, mais quantifier la sensibilité des applications à ces *bottlenecks*. Les applications étudiées ici sont volontairement simples et ont toutes un comportement déterministe : leur contexte d'exécution influence leur vitesse de progression mais pas du tout ce qu'elles font. Le comportement de ces applications est a priori bien capturé par une suite de *phases*, une phase étant un moment de l'exécution de l'application pendant lequel l'application garde le même comportement. Par exemple on s'attend à ce qu'un code qui parcourt 1 Go de RAM pour y compter le nombre d'octets ayant la valeur `4` fasse en boucle un accès mémoire d'un bloc assez gros suivi de calculs locaux au CPU à l'intérieur de ce bloc. L'accès RAM de l'application est fortement limité par la bande passante entre RAM et CPU, alors que le calcul local devrait y être insensible. Cela ne veut pas dire que la progression de la phase de calcul est indépendante de tout bottleneck, on s'attend typiquement à pouvoir observer des ralentissements selon l'utilisation des caches du CPU.

Vous ferez un état de l'art sur la modélisation des machines et des applications. On s'attend à ce que cet état de l'art recouvre à la fois des méthodes à grain fin (centrées sur l'architectures des processeurs) comme celles de OTAWA [1], mais également des méthodes à plus gros grain comme celles de SimGrid [2]. Par exemple la thèse de Idriss Daoudi [3] modélise des effets NUMA en utilisant SimGrid.

Il est attendu que le cœur de votre travail soit réalisé de manière itérative : proposer un modèle (ensemble de paramètres représentant la machine et les applications, ainsi qu'un algorithme permettant d'estimer le taux de progression de chaque application dans une situation donnée), implémenter le modèle dans un langage rapide pour le prototypage de votre choix, évaluer l'intérêt et les limites de ce modèle, conclure et boucler. Des rapports informels sous la forme d'un mini-article seront demandés à chaque itération. Si ces itérations sont concluantes, vous implémenterez un simulateur utilisant SimGrid puis comparerez la qualité des prédictions fournies par votre implémentation *ad hoc* et votre implémentation utilisant SimGrid.

Bibliographie

[1] C. Ballabriga, H. Cassé, C. Rochange, et P. Sainrat, « OTAWA: An open toolbox for adaptive WCET analysis », in *IFIP International Workshop on Software Technolgies for Embedded and Ubiquitous Systems*, 2010, p. 35-46.

[2] H. Casanova, A. Giersch, A. Legrand, M. Quinson, et F. Suter, « Lowering entry barriers to developing custom simulators of distributed applications and platforms with SimGrid », *Parallel Computing*, vol. 123, p. 103-125, 2025, doi: https://doi.org/10.1016/j.parco.2025.103125.

[3] I. Daoudi, « Modélisation de performance et simulation d'applications OpenMP », 2021. [En ligne]. Disponible sur: https://theses.hal.science/tel-03416335

Millian Poquet
Maître de conférences
Univ. Toulouse, IRIT, équipe Sepia
IRIT2-469, 120 cours Rose Dieng-Kuntz, 31400 Toulouse
millian.poquet@utoulouse.fr

Université de Toulouse

Licences, droit d'auteur, propriété intellectuelle

Les implémentations réalisées seront faites sous licence libre ; en Apache-2.0 si possible ou dans une autre licence libre si le projet d'origine en impose déjà une. Les données issues de l'expérience, les documentations, rapports et articles seront écrits sous licence libre CC-BY. Un document de cession de droit d'auteur à l'UT sera signé pour pérenniser la liberté d'accès et de modification de vos productions.

Candidature

Pour maximiser vos chances de candidature, contactez-moi par mail avec les informations suivantes.

- Très courte motivation par rapport au sujet (2 phrases)
- Bulletins de notes (master et licence)
- CV court (parcours d'études, expériences professionnelles, compétences)

Si vous candidatez en groupe à un projet, merci de ne m'envoyer qu'un seul mail pour tout le groupe.