Laboratoire d'Informatique de Grenoble
UMR 5217 - CNRS, INRIA, UGA

Phone : (+33) 4 76 61 53 94
E-mail : mailto :arnaud.legrand@imag.fr
Web : https://team.inria.fr/polaris/members/arnaud-legrand/

Address : Inria Rhône-Alpes,
655 Av. de l'Europe,
Montbonnot St Martin,
38334 St Ismier Cedex
FRANCE

Mise en place de tests de non régression de performance

► **Encadrants :** Vincent Danjean (MdC UGA, LIG/POLARIS) et Arnaud Legrand (CR CNRS, LIG/POLARIS)

► **Compétences souhaitées :**
- UNIX, langages de scripts, shell, ssh, git
- La familiarité avec le langage R est un plus

Quand on développe du code logiciel, il est de bon ton de mettre en place des tests de non régression qui permettent de s'assurer que, quand on modifie le code, on n'introduit pas de bug. Par exemple, dans SimGrid (équipes POLARIS, AVALON, MYRIADS ,...), il y a plus de 500 tests (exécution du soft avec une certaine entrée, récupération de la sortie et comparaison avec la sortie attendue). Le code est recompilé et testé à chaque commit ou bien chaque nuit et l'ensemble de ces informations est collecté dans un outil d'intégration continue comme Jenkins (https://ci.inria.fr/simgrid). De tels mécanismes ont été mis en place pour de nombreuses autres applications comme cado (équipe CARAMEL), une application de cryptographie (https://ci.inria.fr/cado/). Ces types de tests de non-régression sont incroyablement précieux car non seulement ils permettent de s'assurer de la portabilité du code sur une grande variété d'architecture, mais ils servent de filet de sécurité quand on décide de modifier en profondeur le code.

Cependant, ces tests ne couvrent que les aspects fonctionnels d'un logiciel, absolument pas les aspects performances. Il est courant de consacrer un effort important à l'optimisation d'un code mais, une fois ces efforts arrêtés, au fil des modifications fonctionnelles liées à d'autres aspects, les performances se dégradent souvent sans que l'on s'en aperçoive. Il existe ici ou là des solutions "maison" qui permettent de tester ce genre de chose de façon systématique. Par exemple, les équipes STORM et HIEPACS développent des piles logicielles pour le calcul numérique intensif permettant d'exploiter efficacement les architectures multi-coeurs/multi-gpus. Pour surveiller l'évolution des performances de leurs codes, ils réalisent chaque nuit des tests de performance résumées dans des graphiques simples : http://starpu.gforge.inria.fr/testing/morse/trunk/morse.html Les développeurs python peuvent se reposer sur des infrastructures comme codespeed alors que les développeurs du noyau linux repèrent les régressions de performance induites par tel ou tel driver à l'aide du framework Phoronix.

C'est un problème assez difficile car les performances dépendent bien sûr du code mais aussi de la machine et de l'environnement expérimental (version du noyau, des drivers, des bibliothèques, des compilateurs, voire du BIOS). La notion de performance n'est pas simple à définir et est relative à la charge injectée dans le système. Dans le cas de starpu/morse, l'évaluation des performance est effectuée pour une taille de matrice fixe et la métrique utilisée est le temps nécessaire à cette factorisation. L'utilisation de matrices plus petites ou plus grosses mettent en valeur des problèmes différents et idéalement, il faudrait tester sur un ensemble de tailles différentes et s'intéresser à la réponse du système en fonction des entrées plutôt qu'à la performance brute.

L'objectif de ce stage est de concevoir un certain nombre de métriques et de plans d'expériences associés permettant d'évaluer l'évolution des performances d'un code. La pile de solveurs MORSE/STARPU et l'infrastructure Grid5000 seront utilisées comme cas d'étude afin de guider ces développements.