Calcul haute performance en Python pour la Science des Données: une vue d'ensemble

Pierrick Bruneau*, Oscar Castro**, Jean-Sébastien Sottet*

*LIST, 5 Avenue des Hauts-Fourneaux, L-4362 Esch-sur-Alzette
prenom.nom@list.lu,
https://www.list.lu
**Université du Luxembourg, 2 Avenue de l'Universite, L-4365 Esch-sur-Alzette
prenom.nom@uni.lu
https://www.uni.lu

Résumé. Python est devenu le langage préférentiel dans les domaines de la Science des Données et de l'Apprentissage Automatique. Cependant, les *data scientists* ne sont pas nécessairement des programmeurs expérimentés. Bien que Python leur permette d'implémenter rapidement leurs algorithmes, pour passer à l'échelle, l'efficacité du calcul devient un souci inévitable. Ainsi, tirer le meilleur parti des capacités de processeurs multi-coeur et de *Graphical Processing Units* (GPU) n'est généralement pas trivial. Dans cet article, nous présentons les principaux résultats d'un récent article de synthèse, conçu comme un document de référence permettant aux praticiens en Science des Données d'apprivoiser la richesse des outils et des techniques disponibles pour le langage Python. Nous mettons un accent particulier sur la détermination des principaux traits et caractéristiques distinctives des contributions dans ce domaine. Ce document peut aider les praticiens de la Science des Données dans leur choix d'outils, et les développeurs d'outils dans l'identification de manques potentiels dans les travaux existants.

1 Introduction

Python joue un rôle fondamental dans la Science des Données et l'Apprentissage Automatique, avec des bibliothèques de grande importance pratique telles que NumPy, Pandas, TensorFlow, et Scikit-learn. Cependant, l'interpréteur par défaut de Python, CPython, est réputé comme relativement lent, incitant au développement de bibliothèques haute performance dans des langages à typage statique tels que C++, Fortran, et CUDA. Les inefficacités inhérentes à CPython sont principalement dues à son système de typage d'objets dynamique et au surcoût d'exécution lors de l'invocation de fonctions de bibliothèque compilées (Ismail et Suh, 2018). Python, particulièrement son implémentation par défaut, CPython, présente des limites dues à son *Global Interpreter Lock* (GIL), impactant son efficacité en termes de *multi-threading* et d'accès concurrent, conduisant à une faible capacité de parallélisation des instructions. En réponse à ces limitations, de nombreux outils et stratégies pour la programmation haute performance en Python ont été proposés. L'objectif de ce document est de fournir une vue organisée