Composabilité des modèles d'exécution

Équipe concace @ inria Bordeaux
Emmanuel.Agullo@inria.fr, Luc.Giraud@inria.fr, Gilles.Marait@inria.fr,
herault@icl.utk.edu

Stage de fin d'étude (M2/3A) - Printemps 2024

1 Contexte

La séparation des préoccupations (*separation of concerns* en anglais), est "un principe de conception visant à segmenter un programme informatique en plusieurs parties, afin que chacune d'entre elles isole et gère un aspect précis de la problématique générale" [1]. Quand les préoccupations sont clairement séparées, les différentes parties du code peuvent être réutilisées, étendues ou modifiées indépendamment des autres [1]. Une telle conception modulaire permet en particulier de focaliser le développement d'une bibliothèque sur son cœur de métier et de s'appuyer sur des bibliothèques tierces pour le reste des tâches.

Largement reconnue comme une bonne pratique et mise en œuvre dans l'ingénierie logicielle en général [1], ce principe est pourtant peu suivi dans le cadre du calcul haute performance (HPC). En effet, d'une part, la quête de performance requiert souvent un très haut niveau de coopération entre les différentes parties du code. D'autre part, la difficulté du déploiement du code sur des super-calculateurs peut décourager l'emploi de trop nombreuses bibliothèques tierces dont l'inter-compatibilité peut être difficile à garantir.

L'équipe-projet `concace` d' `inria` Bordeaux s'intéresse à la composabilite numérique et parallèle en recourant aux principes de séparation de préoccupations dans un cadre HPC, et plus précisément pour le traitement de systèmes linéaires de grande taille. Elle développe le code `compose` (voir documentation pdf) à cette fin.

Le projet `compose` consiste en l'implantation d'une grande bibliothèque de composants logiciels à haut niveau d'abstraction, afin d'explorer un grand nombre de possibilité combinatoires pour obtenir la meilleure performance numérique et la plus grande efficacité parallèle. À la base du projet se trouve `maphys++`, un solver d'algèbre linéaire hybride qui suit cette philosophie, en pouvant combiner différentes bibliothèques optimisées, notamment des solvers directs (`mumps`, `qr_mumps`, `pastix`...) et des solvers itératifs (gradient conjugué, GMRES, GCR...). Cette approche ayant démontré sa flexibilité et son efficacité, `maphys++` est voué à évoluer en `compose`, en ajoutant des composants logiciels plus variés et en étendant ainsi la gamme de fonctionnalités de la bibliothèque.

2 Objectif

Actuellement, `compose` embarque son propre moteur d'exécution minimaliste basé sur `mpi`, cf. [2]. L'objectif du stage est de développer un nouveau modèle d'exécution à base de tâches [3] pour `compose` en s'appuyant sur le moteur d'exécution `parsec` et son API C++ Template Task Graph (TTG) plus particulièrement [4].

De surcroît, il s'agira de séparer clairement le modèle de programmation de `compose` de son modèle d'exécution de telle sorte que, sans qu'il ne soit nécessaire de changer la programmation des algorithmes numériques, `compose` puisse in fine recourir aussi bien au moteur pré-existant basé sur `mpi` qu'au moteur d'exécution à base de tâches qui sera créé pendant le stage.

3 Cadre du stage

Le stage s'effectuera à l'Inria Bordeaux au sein de l'équipe-projet `concace` en collaboration avec l'Université du Tennessee qui développe le moteur d'exécution `parsec` et co-développe l'API C++ Template Task Graph (TTG).

4 Poursuite en thèse

Le stage pourra se **poursuivre en thèse** au sein de l'équipe `concace`.

5 Notes

Mots-clés: modèle d'exécution, moteur d'exécution, composabilité, expression parallèle

Compétences requises: parallélisme, algorithmique, C++20.

Références:

[1] `https://fr.wikipedia.org/wiki/S%C3%A9paration_des_pr%C3%A9occupations`

[2] `https://solverstack.gitlabpages.inria.fr/maphys/maphyspp/master/#org3625ea9`

[3] Emmanuel Agullo, Olivier Aumage, Mathieu Faverge, Nathalie Furmento, Florent Pruvost, Marc Sergent, Samuel Thibault. Scalability of a task-based runtime system for dense linear algebra applications. `https://hal.inria.fr/hal-01618526/file/tpds14.pdf`

[4] G. Bosilca, R. J. Harrison, T. Herault, M. M. Javanmard, P. Nookala and E. F. Valeev, "The Template Task Graph (TTG) - an emerging practical dataflow programming paradigm for scientific simulation at extreme scale," 2020 IEEE/ACM Fifth International Workshop on Extreme Scale Programming Models and Middleware (ESPM2), 2020, pp. 1-7, doi: 10.1109/ESPM251964.2020.00011.