Laboratoire de l'Informatique du Parallélisme
(UMR CNRS - ENS Lyon - UCB Lyon 1 - INRIA 5668)

équipe ROMA

sujet de stage de Licence 3

Ordonnancement orienté mémoire dans le runtime StarPU

Loris Marchal, Samuel Thibault

Mots-Clés : Calcul parallèle, ordonnancement, algorithmique.

Encadrants :

Loris Marchal : chargé de recherche, CNRS & Univ. de lyon (loris.marchal@ens-lyon.fr)
Yves Robert : professeur des universités, ENS Lyon & Univ. de lyon (loris.marchal@ens-lyon.fr)
Samuel Thibault : Maitre de conférence, Université de Bordeaux & INRIA, en délégation dans l'équipe ROMA en février–juillet 2019 (samuel.thibault@inria.fr)

équipe d'accueil

Le stage se déroulera dans l'équipe ROMA, du Laboratoire de l'Informatique du Parallélisme. L'équipe ROMA s'intéresse à la conception d'algorithmes parallèles et d'ordonnancement pour les plates-formes de calcul distribuées, en particulier pour les applications scientifiques.

Contexte du stage

Les plates-formes de calcul parallèle, utilisées par exemple pour effectuer des simulations numériques de grandes tailles, deviennent de plus en plus complexes : utilisation d'accélérateurs de calcul, hiérarchies mémoires profondes, combinaison de mémoires partagées et distribuées...Pour tirer partie de leur puissance, il devient nécessaire de s'appuyer sur des ordonnanceurs légers, tels que le runtime StarPU [1], développé (entre autres) par Samuel Thibault. Une application de calcul est alors décrite comme un graphe de tâches, dont les arêtes modélisent les dépendances, et l'ordonnanceur s'efforce de placer les tâches disponibles sur les différents cœurs de calcul pour optimiser les performances. Entre autres problèmes à résoudre pour bien ordonnancer un tel graphe se pose la question de la mémoire : il faut à tout prix éviter la situation de pénurie de mémoire, au risque de voir l'exécution échouer (ou tout au moins obtenir des performances largement dégradées).

Des solutions à ce problème ont été proposées. Tout d'abord, des ordonnanceurs prenant en compte la mémoire ont été proposés [5]. Plus récemment, nous avons proposé une méthode qui ajoute des dépendances au graphe de tâches afin que toute exécution ne dépasse pas une quantité de mémoire donnée [6]. Cette méthode a le grand avantage de ne pas brider l'ordonnanceur plus que nécessaire, afin qu'il puisse s'adapter aux conditions réelles au moment de l'exécution. Malheureusement, la complexité de la solution et le grand nombre de dépendances ajoutées rend cette méthode peu utilisable en pratique.

Objectif et déroulement

L'objectif de ce stage est d'améliorer cette méthode, en limitant les garanties à fournir : au lieu de s'assurer que toute exécution ne dépasse pas la mémoire, il suffit en effet de garantir l'absence de situations d'interblocage (deadlock) à cause la mémoire : les runtimes tels que StarPU savent bloquer une tâche demandant de la mémoire tant que celle-ci n'est pas disponible. Ce méchanisme devrait permettre de rajouter bien moins de nouvelles dépendances au graphe.

Le travail demandé pendant se stage consistera en plusieurs étapes.

1. Modéliser le problème sous forme d'un problème de graphe, en choisissant la modélisation mémoire qui paraît la plus pertinente et en exprimant l'objectif poursuivi comme une propriété du graphe.

2. Proposer des solutions algorithmiques pour résoudre ce problème, en cherchant à réduire la complexité de ces solutions.
3. Tester l'efficacité des solutions proposées d'abord par une campagne de simulations sur des graphes connus.
4. Implanter certaines de ces solutions dans le runtime StarPU.

La répartition du travail entre la théorie et la pratique pourra être adaptée en fonction des compétences et des attentes du candidat.

Compétences requises

Le candidat devra avoir un bon niveau en algorithmique, à l'aise avec la programmation dans divers langages communs (C, C++). Un bon niveau en anglais est un plus.

Apport pour le stagiaire

Le stagiaire aura l'opportunité d'intégrer un groupe de recherche dynamique, d'acquérir des connaissances dans le domaine de l'ordonnancement et du calcul parallèle, d'accroître son expertise en programmation et dans le développement des outils les plus récents pour le calcul parallèle.

Bibliographie

[1] Cédric Augonnet, Samuel Thibault, Raymond Namyst, and Pierre-André Wacrenier. StarPU : a unified platform for task scheduling on heterogeneous multicore architectures. *Concurrency and Computation : Practice and Experience*, 23(2) :187–198, 2011.

[2] Lionel Eyraud-Dubois, Loris Marchal, Oliver Sinnen, and Frédéric Vivien. Parallel scheduling of task trees with limited memory. *TOPC*, 2(2) :13 :1–13 :37, 2015.

[3] P. Hénon, P. Ramet, and J. Roman. PaStiX : A High-Performance Parallel Direct Solver for Sparse Symmetric Definite Systems. *Parallel Computing*, 28(2) :301–321, January 2002.

[4] Joseph Y.-T. Leung, editor. *Handbook of Scheduling - Algorithms, Models, and Performance Analysis*. Chapman and Hall/CRC, 2004.

[5] Joseph W. H. Liu. An application of generalized tree pebbling to sparse matrix factorization. *SIAM J. Algebraic Discrete Methods*, 8(3) :375–395, 1987.

[6] Loris Marchal, Hanna Nagy, Bertrand Simon, and Frédéric Vivien. Parallel scheduling of dags under memory constraints. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 204–213, 2018.

[7] Alex Pothen and Chunguang Sun. A mapping algorithm for parallel sparse cholesky factorization. *SIAM Journal on Scientific Computing*, 14(5) :1253–1257, 1993.

[8] Yves Robert. Task graph scheduling. In *Encyclopedia of Parallel Computing*, pages 2013–2025. Springer, 2011.