# Programmation des architectures hétérogènes à l'aide de tâches divisibles ou moldables

Abdou Guermouche and Pierre-André Wacrenier and Raymond Namyst
INRIA « Runtime » and « HIEPACS » groups
Contact Email: Raymond.Namyst@labri.fr

Mots clés : Calcul parallèle, Accélérateur, GPU, multicoeur, StarPU.

## Contexte et objectifs du travail

Les ordinateurs multicoeurs équipés d'accélérateurs réalisent une percée remarquable dans le paysage du calcul haute performance. Parmi les machines parallèles les plus puissantes au monde (selon le classement www.top500.org), une sur deux est équipée d'accélérateurs. Cette récente évolution vers des architectures hétérogènes a entraîné un regain d'efforts de recherche visant à concevoir des outils permettant de programmer facilement des applications capables d'exploiter efficacement toutes les unités de calcul de ces machines.

Le support d'exécution StarPU, développé dans l'équipe Runtime, a été conçu pour servir de cible à des compilateurs de langages parallèles et des bibliothèques spécialisées (algèbre linéaire, développements de fourier, etc.) La fonction principale de StarPU est d'ordonnancer des graphes dynamiques de tâches de manière efficace sur l'ensemble des ressources hétérogènes de la machine. Pour ce faire, le support s'appuie sur des modèles adaptatifs de prédiction de coût des calculs et des transferts de données, ainsi que sur une mémoire virtuellement partagée destinée à minimiser les mouvements de données entre les différentes mémoires. StarPU est avant tout une plateforme pour expérimenter de nouvelles stratégies d'ordonnancement, celles-ci pouvant aisément être construites en redéfinissant les fonctions appelées en réaction à certains événements (nouvelle tâche prête, unité de calcul oisive, etc.) StarPU a récemment été utilisé avec succès pour l'implémentation d'algorithmes parallèles en algèbre linéaire sur configurations multi-GPU, en collaboration avec d'autres équipes françaises et étrangères.

L'un des aspects les plus difficiles, lors du découpage d'une application en graphe de tâches, est de choisir la granularité de ce découpage, qui va typiquement de pair avec la taille des blocs utilisés pour partitionner les données du problème. Les granularités trop petites ne permettent pas d'exploiter efficacement les accélérateurs de type GPU, qui ont besoin de mettre en oeuvre un parallélisme massif pour « tourner à plein régime ». À l'inverse, les processeurs traditionnels exhibent souvent des performances optimales à des granularités beaucoup plus fines. Le choix du grain d'un tâche dépend non seulement du type de processeur sur lequel elle s'exécutera, mais il a en outre une influence sur la quantité de parallélisme disponible dans le système : trop de petites tâches risque d'inonder le système en introduisant un surcoût inutile, alors que peu de grosses tâches risque d'aboutir à un déficit de parallélisme. Le découpage d'une application en graphe

de tâches peut donc être lui même adaptatif, dépendant de l'ordonnancement et de l'architecture cible.

Pour appréhender ce problème de granularité deux approches complémentaires seront considérées. La première consiste à agréger des ressources de manière à obtenir la performance optimale pour une opération donnée. Le problème est alors de mettre en œuvre les mécanismes de gestion de tâches parallèles au sein des supports d'exécution et de considérer les problèmes d'ordonnancement associés (modèles de tâches malléables et moldables). Un deuxième axe est d'explorer la notion de tâches divisibles, c'est-à-dire de tâches que le support d'exécution pourra décider (ou non) de redécouper en plusieurs sous-tâches à l'exécution, et d'étudier de nouvelles stratégies d'ordonnancement adaptant la granularité des calculs en fonction de différents critères tels que la quantité de parallélisme que l'on souhaite générer, l'opportunité d'exploiter certains types d'unités de calcul à un moment donné, etc. Une partie du travail consistera à étudier comment gérer efficacement une partition non uniforme des données (pour autoriser la co-existence de sous-données de différentes granularités) ainsi qu'une gestion des dépendances entre tâches s'adaptant à des raffinements locaux du graphe de tâches.

La validation de cette étude se fera dans le cadre de l'optimisation de l'exécution d'applications complexes telles que des application de simulation numériques (application de mécanique des fluides, etc), solveurs linéaires creux etc. Il est à noter que cette thèse se déroulera dans le contexte du projet ANR SOLHAR (ANR-13-MONU-007) impliquant en plus de l'équipe-projet Inria Runtime des spécialistes en ordonnancement, en solveurs linéaires ainsi que des partenaires industriels. Ceci permettra au doctorant de se trouver dans un contexte collaboratif qui pourra lui permettre d'appréhender les différents aspects de la problématique de l'amélioration des performances des applications complexes sur plates-formes modernes.

## Bibliographie


— Cédric Augonnet, Samuel Thibault, Raymond Namyst, and Pierre-André Wacrenier. StarPU : A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures. In Proceedings of the 15th International Euro-Par Conference, volume 5704 of Lecture Notes in Computer Science, Delft, The Netherlands, pages 863-874, August 2009.

— Emmanuel Agullo, Cédric Augonnet, Jack Dongarra, Mathieu Faverge, Hatem Ltaief, Samuel Thibault, and Stanimire Tomov. QR Factorization on a Multicore Node Enhanced with Multiple GPU Accelerators. In 25th IEEE International Parallel & Distributed Processing Symposium (IEEE IPDPS 2011), Anchorage, Alaska, USA, May 2011.

— Alexandre Duchateau, David Padua, Denis Barthou. Hydra : Automatic Algorithm Exploration from Linear Algebra Equations, in "ACM/IEEE Intl. Symp. on Code Optimization and Generation", Shenzhen, China, IEEE Computer Society, Feb. 2013.


## Liens utiles

— Équipe Runtime : runtime.bordeaux.inria.fr
— Logiciel StarPU : runtime.bordeaux.inria.fr/starpu