Programmation des architectures hétérogènes à l'aide de tâches hiérarchiques

A. Guermouche - R. Namyst - P.A Wacrenier

Contexte

Les ordinateurs multicoeurs équipés d'accélérateurs réalisent une percée remarquable dans le paysage du calcul, tant au niveau de la station de travail qu'à celui des machines parallèles les plus puissantes au monde. Cette 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.

Dans ce contexte l'Équipe SATANAS développe depuis 2008 un support d'exécution nommé StarPU (http ://starpu.gforge.inria.fr) capable d'ordonnancer des ensembles de tâches sur des machines multicoeur dotées de plusieurs accélérateurs graphiques (machines multi-GPU) ou d'accélrateurs tel le Xeon-Phi. L'efficacité de StarPU réside dans l'utilisation de modèles de coût permettant de prédire les performances des tâches en fonction du processeur cible, ainsi que dans l'utilisation d'un gestionnaire de mémoire virtuellement partagée permettant de minimiser les transferts de données entre accélérateurs. StarPU est le logiciel référence du domaineet est utilisé dans des centres de recherche académiques et industriels pour faire du calcul scientifique ou de la recherche en calcul haute performance.

Le paradigme des tâches hiérarchiques

L'objectif de ce projet est d'introduire dans StarPU la notion de tâche hirérarchique, c'est-à-dire de tâche que le support d'exécution transforme en un sous-graphe de tâches à l'exécution, et d'étudier de nouvelles stratégies d'ordonnancement adaptant le calcul 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.

L'étude de ce modèle nous motive particulièrement car il porteur de solutions à quelques écueils rencontrés par les utilisateurs de StarPU :

1. La soumission séquentielle de tâches. Pour des raisons de simplicité de programmation les utilisateurs de StarPU ont pour usage de soumettre de façon séquentielle les graphes de tâches ce qui peut engendrer des pertes de performances. Les tâches hiérarchiques permettent contourner cette difficulté, en rendant possible la soumission parallèle de tâches tout en conservant le même confort de programmation.
2. La gestion des ressources critiques. Les utilisateurs avancés de StarPU ont pour usage de réguler la soumission de tâches pour limiter / contrôler la consommation de ressources critiques (mémoire, communication réseau). Cette technique peut pénaliser les performances (déficit de parallélisme) et n'est pas adapdée à la gestion concurente de plusieurs type de ressources. L'utilisation de tâches hiérarchiques permet de recentrer cette gestion au niveau du moteur d'ordonnancement - il est alors possible d'implémenter des algorithmes de gestion de ressource de type Banquier de Dijkstra.
3. La gestion adaptative de la granularité. 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'une 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.

Objectifs des stages

Nous venons d'implémenter et de valider les briques de base nécéssaires à la création et à l'exécution des tâches hiérarchiques au sein de StarPU. Cette implémentation permet la soumission parallèle de tâches de calcul. Les stages proposés portent sur les deux derniers points, à savoir :

Gestion des ressources critiques et expression du parallélisme Il s'agit d'implémenter au sein de StarPU un ordonnanceur générique permettant d'appliquer des politiques classiques de gestion de ressources. Dans un second temps on se concentrera sur la gestion mémoire en appliquant des techniques présentées et simulées dans [5].

Gestion adaptative de la granularité et ordonnancement Il s'agit de proposer une interface de programmation facilitant au sein de StarPU la gestion de la granularité tout d'abord dans un cadre statique (ie. la granularité de chaque tâche hiérarchique est imposée par l'application) puis dans un cadre dynamique. Pour ce dernier point, on pourra s'inspirer des techniques présentées et simulées dans [6].

Bibliographie

1 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.

2 Emmanuel Agullo, Alfredo Buttari, Abdou Guermouche, and Florent Lopez. Implementing multifrontal sparse solvers for multicore architectures with Sequential Task Flow runtime systems. ACM Transactions On Mathematical Software, 2016.

3 W. Wu, A. Bouteiller, G. Bosilca, M. Faverge, and J. Dongarra, "Hierarchical dag scheduling for hybrid distributed systems," in 29th IEEE International Parallel & Distributed Processing Symposium (IPDPS), Hyderabad, India, May 2015.

4 Terry Cojean, Abdou Guermouche, Andra Hugo, Raymond Namyst, Pierre-André Wacrenier. Resource aggregation in task-based applications over accelerator-based multicore machines (HeteroPar'2016 worshop of Euro-Par, Aug 2016, Grenoble, France. 2016)

5 Guillaume Aupy, Clément Brasseur and Loris Marchal. « Dynamic memory-aware task-tree scheduling » https ://hal.inria.fr/hal-01390107

6 Antón Rey, Francisco D. Igual , Manuel Prieto-Matía, « HeSP : A Simulation Framework for Solving the Task Scheduling-Partitioning Problem on Heterogeneous Architectures » Euro-Par 2016 : Parallel Processing Volume 9833 of the series Lecture Notes in Computer Science pp 183-195