Exécution parallèle de code de combustion à l'aide du paradigme master/worker

Hakim Oueslati*[1], Laurent Bobelin[2], and Mame William-Louis[3]

[1]Laboratoire dÍnformatique Fondamentale dÓrléans – Université d'Orléans : EA4022, Institut National des Sciences Appliquées - Centre Val de Loire – France

[2]Laboratoire dÍnformatique Fondamentale dÓrléans (LIFO) – Université d'Orléans : EA4022, Institut National des Sciences Appliquées - Centre Val de Loire : EA4022 – Bâtiment IIIA, Rue Léonard de Vinci, B.P. 6759, F-45067 ORLEANS Cedex 2, France

[3]Laboratoire PRISME (PRISME) – Université d'Orléans : EA4229 – 63 Avenue de Lattre de Tassigny, 18000 Bourges, France

Résumé

De nombreuses applications scientifiques et industrielles nécessitent une puissance de traitement bien supérieure à celle d'une station de travail. La dynamique des fluides numérique (CFD) est très exigeante en termes de mémoire et de calculs. De nos jours, la simulation de cas réalistes peut se compter en semaines, voire en mois. Poussées par la présence de superordinateurs à haute vitesse, les approches de parallélisation ont été plus utilisées afin d'obtenir de meilleurs résultats et de résoudre les problèmes les plus complexes. Dans cet article, nous introduirons une approche basée sur les tâches où on étudie des possibilités offertes par le paradigme de la distribution de la charge master/worker afin de paralléliser un code en déployant une équipe de threads (y compris un thread de gestion de l'application) et d'atteindre de meilleures performances. Nous développerons un framework de programmation générique afin d'acquérir la possibilité d'appliquer la méthode sur n'importe quelle application. Cette approche nous permet aussi de déporter les calculs dans une enclave SGX afin de protéger le code de simulation et les données.
En raison de la complexité du code principal METAS, un code de pile thermique a été traité afin de vérifier l'efficacité de l'approche basée sur les tâches. Enfin, nous présenterons deux méthodes d'implémentation de la gestion des tâches (ordonnancement et exécution) et quelques résultats afin de démontrer les avantages de la méthode et d'ouvrir quelques perspectives.

Mots-Clés: master/worker, ordonnancement des tâches, threads

*Intervenant