Accélération GPU de simulation numérique d'atomisation de jet

Loïc Reynier*[1], Davide Zuzio[2], Jean-Christophe Hoarau[3], and Jean-Luc Estivalèzes[4]

[1]DMPE, ONERA, Université de Toulouse [Toulouse] – ONERA, CNES – France
[2]DMPE, ONERA, Université de Toulouse [Toulouse] – ONERA – France
[3]ONERA – ONERA – France
[4]DMPE, ONERA, Université de Toulouse [Toulouse] – ONERA – France

Résumé

Motivation

Le code de recherche DyJeAt, développé à l'ONERA, est utilisé pour la simulation numérique directe d'écoulements diphasiques incompressibles. Il a notamment permis de réaliser une simulation d'atomisation à très grande échelle, comptant plus de 3 milliards de cellules, sur 86 400 cœurs mobilisant 40 Mh de calcul.

Dans le cadre d'un projet de recherche en collaboration avec le CNES sur les instabilités de combustion dans les moteurs de fusée, des simulations d'une complexité et d'une résolution similaire doivent être reproduites de manière plus systématique.

Dans le but d'accélérer le code, un portage GPU du code a ainsi été initié dans le cadre d'un post-doctorat débuté en octobre 2024. L'objectif de cette présentation est de partager les résultats obtenus en termes d'accélération, ainsi qu'un retour d'expérience technique sur le portage GPU d'un code CFD complexe.

Travail technique et scientifique

Nous avons adopté une approche OpenACC pour l'ensemble du code, dans le but de limiter l'intrusion dans la base existante, écrite en Fortran 90/2003. Cette approche permet non seulement de déployer et paralléliser les boucles de calcul sur les GPU sans avoir à réécrire les routines dans un langage spécifique tel que CUDA, mais offre également une portabilité importante. En effet, OpenACC est compatible avec plusieurs marques de GPU (NVIDIA, AMD, Intel), ce qui facilite le déploiement du code sur différentes architectures sans

*Intervenant

modification majeure.
L'essentiel du travail a consisté à annoter les boucles de calcul avec des directives OpenACC, ainsi qu'à
adapter les communications MPI pour permettre les échanges inter-GPU de manière performante.
Le solveur de pression du code, basé sur un gradient conjugué multigrille, représente la partie la plus
critique en termes de ressources. Il a été entièrement porté en OpenACC. En parallèle, une seconde implémentation s'appuyant sur la bibliothèque de solveurs multigrille AMGX de NVIDIA a été développé, dans
l'objectif d'évaluer si cette solution pouvait offrir de meilleures performances et une meilleure précision sur
des machines équipées de GPU NVIDIA.
Les choix d'implémentation, les difficultés rencontrés, ainsi que les performances obtenues avec les deux
implémentations seront détaillées dans la présentation.
Une analyse paramétrique des configurations proposées par AMGX a été menée ; elle a montré que, pour
notre configuration, le solveur le plus performant est un gradient conjugué préconditionné par un multigrille
par agrégation. Cette étude comparative sera également présentée.
Le portage GPU du bloc résolvant le reste des équations de Navier-Stokes est en cours d'achèvement.
Une fois celui-ci terminé, il ne restera plus qu'à porter le module de suivi d'interface. Cette dernière étape
devrait être plus rapide, dans la mesure où elle repose sur des structures de communication similaires à celles
déjà traitées - les communications étant le point chronophage du portage GPU du code.

Originalité

L'originalité de ce travail réside dans la capacité à obtenir une accélération significative sur GPU, avec une
bonne scalabilité multi-nœud, sur un code de mécanique des fluides diphasique complexe, sans avoir à en
refondre entièrement la structure. Si l'accélération GPU est aujourd'hui une approche courante, notamment
dans le domaine de l'IA, atteindre des performances satisfaisante sur un code de CFD diphasique – fortement
couplé, parallèle et numériquement exigeant - reste un défi non trivial.

Résultats

Les mesures de performances réalisées sur le solveur de pression, à nombre de nœuds équivalent, montrent un
facteur d'accélération GPU compris entre 5 et 12 selon les configurations et les machines. Cette amélioration
s'accompagne d'une consommation électrique réduite d'un facteur 2 à 3, ce qui souligne l'intérêt de l'approche
non seulement en termes de performance, mais également d'efficacité énergétique.
Par ailleurs, un strong scaling jusqu'à 10 GPU a été réalisé sur un cas réaliste, montrant un speedup
proportionnel au nombre de GPU, ce qui confirme la bonne scalabilité multi-nœud.

Ressources utilisées

Le développement ainsi que les mesures de performances et de scalabilité ont été réalisées

sur plusieurs
calculateurs possédant des nœuds GPU : Turpan (CALMIP, GPU NVIDIA A100), Topaze (CCR, GPU
NVIDIA A100) et Olympe (CALMIP, GPU NVIDIA V100).

Références

J.-C. Hoarau, L.-H. Dorey, D. Zuzio, F. Granger, J.-L. Estivalèzes, Direct numerical simulation of a subcritical coaxial injection in fiber regime using sharp interface reconstruction, International Journal of Multiphase Flow.