





Etudiante-es :
Barras Simon

Résumé travail de Bachelor 2023
INFORMATIQUE ET SYSTÈMES DE COMMUNICATION

PROFESSEUR-ES : Frederic Bapst, Jean Hennebert	
MANDANT : Lawrence Berkeley National Laboratory, Paolo Calafiura, Julien Esseiva	
CONTACT INTERNE : iCoSys	
ACRONYME DU PROJET : GPU Optimization Celeritas	NO INTERNE OU FILIÈRE : B23ISC27
ORIENTATION : Informatique logiciel	EXPERT-ES : Dr. Baptiste Wicht
OBJECTIFS DE DEV. DURABLE :  	

Optimisation GPU dans Celeritas, une bibliothèque de simulation de détecteurs pour la physique haute énergie.

Les détecteurs du LHC du CERN produisant de plus en plus de données, les simulations pour les expériences ATLAS et CMS ne peuvent plus se faire avec des outils uniquement CPU. L'arrivée des GPU dans les supercalculateurs ouvre de nouvelles perspectives de développement. C'est ce qu'exploite l'équipe Celeritas, pour fournir une bibliothèque de simulations de physique des particules accélérées par les GPU.

Pour son travail de Bachelor, Simon Barras s'est rendu à Berkeley, aux États-Unis, pour travailler au Lawrence Berkeley National Laboratory. C'est là qu'il a collaboré avec l'équipe Celeritas, qui est répartie dans plusieurs laboratoires à travers les États-Unis, notamment à Oak Ridge.

Celeritas

Ce projet est une bibliothèque de transport de particules Monte-Carlo pour la simulation des détecteurs de physique des hautes énergies. Il est motivé par les besoins massifs de calcul des expériences ATLAS et CMS dans le LHC au CERN. L'objectif est de remplacer Geant4 qui est l'outil historique pour la simulation des particules. Celeritas est développé pour être lancé sur des supercalculateurs.

GPU

Les processeurs graphiques (GPU) sont des puces définies comme étant à instructions uniques et données multiples (SIMD) ou à instructions multiples et threads multiples (SIMT). Ces cartes

sont efficaces lorsqu'un petit ensemble d'opérations doit être effectué sur une grande échelle de données.

Pour développer un programme à l'aide d'un GPU, la plateforme du GPU est nécessaire pour définir quelle partie du code doit être exécutée sur l'appareil. Par exemple, la plateforme de Nvidia est appelée CUDA. Lorsque nous lançons un noyau, nous définissons le nombre de blocs et le nombre de threads par bloc.

Dormand Prince

Runge Kutta Dormand Prince (RKDP) est un solveur d'équations différentielles ordinaires du quatrième et du cinquième ordre. Il est défini par le coefficient du tableau de Butcher. L'implémentation se fait en neuf étapes. Les six premières étapes consistent à calculer l'étape intermédiaire à l'aide de l'équation et à mettre à jour l'état actuel. Les trois dernières étapes consistent principalement à calculer l'erreur et l'état du point résultant.

Conception

Il est important de bien comprendre le code pour trouver un moyen de le paralléliser, même si la méthode RKDP est itérative et que cela semble impossible.

L'implémentation de la méthode dans le projet comporte trois types de code. La première catégorie est le calcul de l'état intermédiaire avec l'équation. La deuxième partie est le code qui n'a pas de dépendances et c'est la multiplication des coefficients par la taille du pas. La dernière catégorie est la mise à jour de l'état qui est un vecteur à deux



Etudiante-es :
Barras Simon

Résumé travail de Bachelor 2023
INFORMATIQUE ET SYSTÈMES DE COMMUNICATION

dimensions. Cette partie peut être divisée en sous-tâches qui peuvent être parallélisées. La figure 1 montre un profil du temps d'exécution pour chaque étape et chaque type.

D'après ces résultats et en supposant que la nouvelle implémentation sera distribuée sur quatre threads, il est possible de distribuer les tâches pour réduire le temps d'exécution de la méthode. La figure 2 montre un exemple de charge de travail optimale.

Implémentation

Deux implémentations ont été réalisées. La première est conçue comme un contrôleur et calcule la partie séquentielle lorsque les trois travailleurs calculent la multiplication vectorielle. Les coefficients ne sont calculés qu'à la volée, car le stockage dans des variables ralentit le processus. La différence entre les deux implémentations réside dans l'utilisation de la mémoire partagée. Cet espace mémoire est plus rapide que la mémoire globale, mais sa taille est limitée à quelques kilooctets par bloc. La figure 3 présente deux diagrammes illustrant la communication au sein des threads.

Résultats

Les résultats des deux implémentations sont assez bons si un seul bloc est utilisé. Elles accélèrent le processus de 150 % en moyenne, mais elles peuvent suivre moins de particules que l'ancienne version. La version à mémoire globale est limitée par le nombre de threads par piste et la version à mémoire partagée est limitée par sa propre taille. La figure 4 montre l'accélération et les limitations.

Comme Celeritas veut suivre un grand nombre de particules, il fonctionne sur plus d'un bloc.

La figure 5 montre l'évolution du temps en fonction du nombre de pistes. Les résultats sont assez mauvais, car la nouvelle version a des difficultés à passer à l'échelle. Chaque fois que 84 blocs sont demandés (environ 15'000 pistes pour les nouvelles implémentations), la quantité de ressources demandées au GPU dépasse ce qu'il peut exécuter à la fois. Ces dépassements sont illustrés comme le montre le saut sur le graphique.

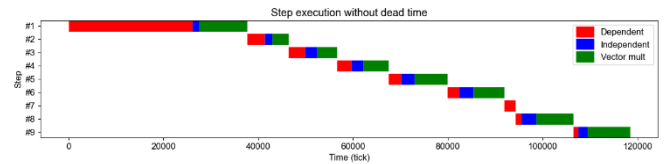


Figure 1 Results of the RKDP profiling

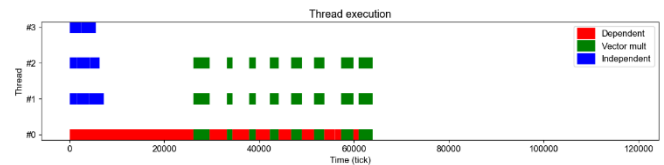


Figure 2 Planification optimale des tâches

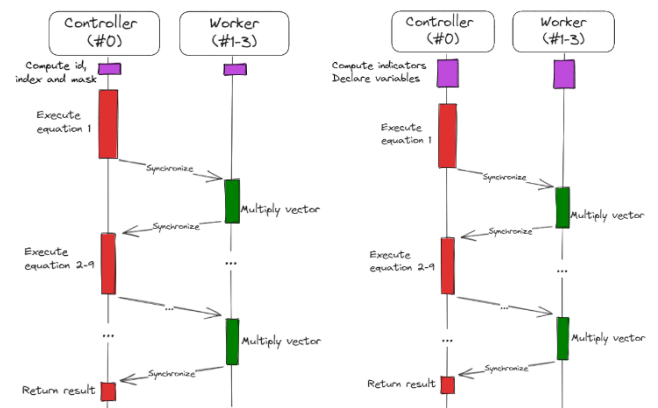


Figure 3 Mise en œuvre avec respectivement une mémoire globale et une mémoire partagée

Evolution of the different implementations with the number of tracks using 1 block

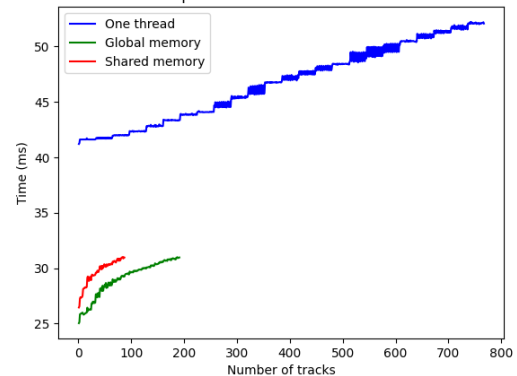


Figure 4 Performance avec un seul bloc

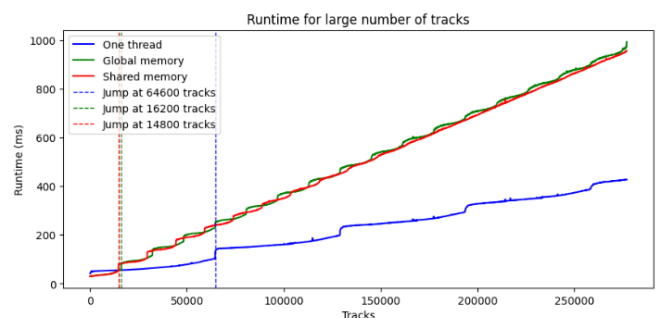


Figure 5 Performance avec plusieurs blocs