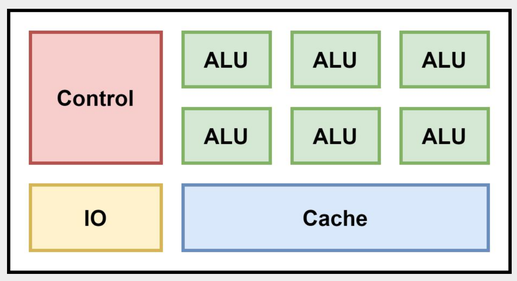
PER2024-004 - Étude des calculs GP-GPU sur cartes graphiques avec CUDA et OpenCl

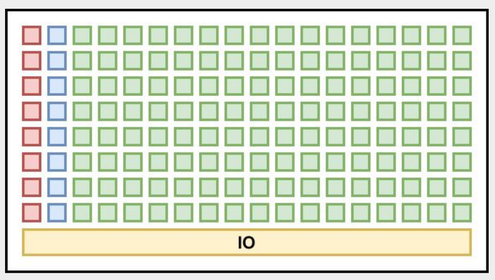
Étudiants (SI5-IoT-CPS) : Quentin Maurois et Julien Soto
Encadrant : Sid Touati
Contacts : quentin.maurois@etu.unice.fr julien.soto@etu.unice.fr sid.touati@univ-cotedazur.fr

1 Contexte

Avec l'**essor** de l'IA, du **calcul scientifique** et du rendu 3D, la **demande en puissance de calcul explose**. Les **GPU**, optimisés pour le **calcul parallèle**, surpassent les **CPU** sur ces tâches, mais leur coût reste élevé. **NVIDIA** domine le marché avec ses RTX, face aux alternatives d'**Intel** et **AMD**. Des **librairies**, comme **OpenCL** et **CUDA**, jouent un rôle clé dans l'**optimisation des calculs lourds**, alliant performance et maîtrise des coûts.

2 CPU Vs GPU





✅Calculs séquentiels

✅Attribution des tâches par le programmeur

✅Optimisé pour la logique complexe et les prises de décision rapides

✅Calculs parallèles

✅Exécution automatique de nombreuses tâches simultanément

✅Optimisé pour le traitement massif de données (IA, rendu 3D, simulations)

3 CUDA Vs OpenCL

✅ Performances optimisées pour **GPU NVIDIA**

✅ Écosystème riche : bibliothèque **outils de profiling** et support

❌**Limité aux GPU NVIDIA**

❌**Fermé et propriétaire**

✅Standard ouvert, supporté sur **GPU**, **CPU**, **FPGA**, et autres

✅Flexibilité et **portabilité** entre différents matériels

❌**API plus complexe** et **moins optimisée** par plateformes

❌**Performances variables** selon les implémentations et matériels

4 Environnement de test

💻 **Core i7-8750H, GTX 1050 Ti – Ubuntu 22.04**

🔧 **Pilotes** selon GPU

🎯**Objectif : Comparer CPU et GPU sur calculs matriciels avec CUDA et OpenCL**

5 Méthodologie

🟢**Matrices carrées** (tailles : puissances de 2, 1 à 8192)

⚡**Opérations** : addition, multiplication matricielle, déterminant, trace, transposition

🖥️ **Comparaison CPU** (C++, C) vs **GPU** (CUDA, OpenCL)

⏱️Mesure du **temps d'exécution** (incluant transferts GPU↔RAM)

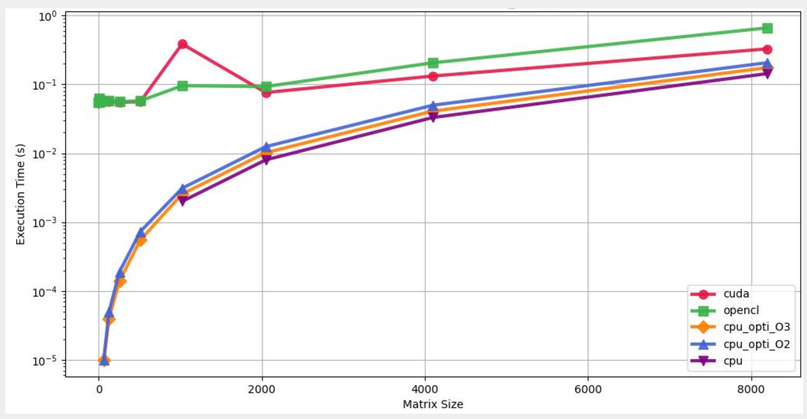
📈**Courbes de performance** pour chaque opération

6 Résultats

Petites matrices (<256) : CPU plus rapide, transfert vers GPU annule l'avantage du parallélisme.

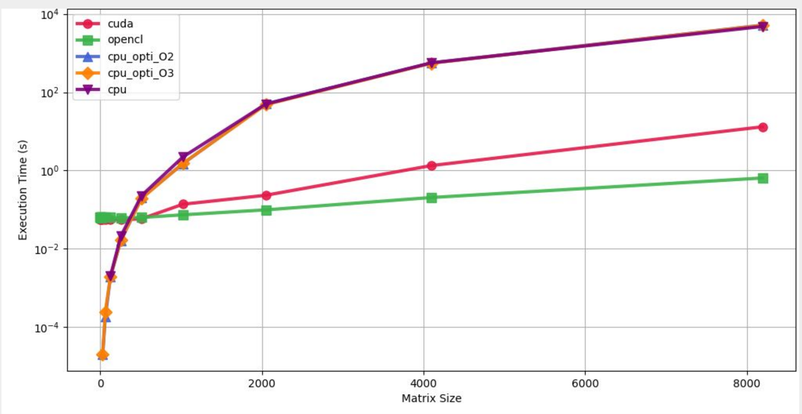
Opérations simples (Addition) : CPU domine, limité uniquement par l'accès mémoire, comme GPU.

Opérations complexes (multiplication) : CPU inefficace dès **256×256**, dépassant **1h** pour **8192×8192**. GPU compense coût initial et exécute calculs en **moins d'1s** grâce au parallélisme.



Temps d'exécution de l'addition matricielle selon la taille

Temps d'exécution de la multiplication matricielle selon la taille



7 Conclusion

Le **GPU** excelle sur les **calculs intensifs** grâce à son **parallélisme massif**, mais plusieurs limitations existent : les **transferts RAM↔VRAM**, gérés par le CPU, sont coûteux, la **synchronisation avec le CPU** peut ralentir l'exécution, et les I/O directes entre la carte graphique et le disque sont impossibles. Le **CPU** reste donc **plus performant pour les tâches légères** et les opérations dépendantes de la mémoire, tandis que le GPU devient indispensable pour les calculs complexes à grande échelle.
Concernant les API**, CUDA**, optimisé pour NVIDIA, n'a **pas** montré un **avantage systématique sur OpenCL** dans nos tests et peut même être moins performant dans certains cas. Cependant, **OpenCL**, bien que plus flexible et compatible avec diverses plateformes, est plus **difficile à implémenter** et optimiser, rendant son **utilisation plus exigeante**.

UNIVERSITÉ CÔTE D'AZUR · Inria · POLYTECH NICE SOPHIA