Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

CEG 4536 Architecture des ordinateurs III
Automne 2024

À remettre 2 novembre 23h 59

Lab2 : Optimisation et Exécution Parallèle avec CUDA C

1. Introduction

Dans ce laboratoire, vous allez explorer et appliquer des concepts avancés de programmation parallèle sur GPU à l'aide de CUDA C. Vous apprendrez à optimiser des algorithmes tout en comprenant les mécanismes sous-jacents tels que la gestion des warps, l'exécution dynamique, et les techniques pour réduire la divergence des threads. Vous utiliserez également l'outil `nvprof` pour profiler votre code, identifier les goulets d'étranglement, et optimiser les performances.

Ce laboratoire est divisé en plusieurs étapes où vous implémenterez, optimiserez et profilerez des algorithmes de réduction parallèle. Vous explorerez également la parallélisation dynamique et l'exécution imbriquée pour comprendre comment exposer plus de parallélisme dans un environnement GPU.

2. Objectif

L'objectif principal de ce laboratoire est d'acquérir une compréhension approfondie du modèle d'exécution CUDA et de maîtriser les techniques d'optimisation suivantes:

- Modèle d'exécution des warps et blocs de threads.
- Réduction des divergences des warps.
- Optimisation des performances par unrolling des boucles et utilisation des templates.
- Exécution imbriquée et parallélisation dynamique.
- Utilisation de `nvprof` pour profiler et optimiser le code.

Les étudiants devront optimiser un algorithme de réduction parallèle tout en minimisant la divergence des threads et en maximisant l'occupation des ressources. Ils utiliseront également `nvprof` pour analyser les performances et explorer la scalabilité du programme.

3. Réalisation

Tâche 1: Introduction au modèle d'exécution CUDA et analyse de la performance avec `nvprof`

1. Implémentez un programme CUDA C qui exécute une réduction parallèle pour additionner les éléments d'un tableau.
2. Utilisez `nvprof` pour mesurer les warps actifs et analyser les opérations mémoire.
3. Identifiez les problèmes de divergence des warps dans votre programme et proposez des améliorations.

Concepts couverts :

- Modèle d'exécution des warps : Comment les warps exécutent les threads parallèles.
- nvprof : Profilage des warps actifs et des opérations mémoire.

Tâche 2: Optimisation des performances et gestion de la divergence des warps

1. Améliorez l'algorithme de réduction en réduisant la divergence des warps.
2. Implémentez les techniques d'unrolling des boucles pour améliorer les performances.
3. Utilisez des fonctions templates pour rendre l'algorithme de réduction plus générique et modulaire.

Concepts couvertes:

- Évitement de la divergence des branches : Minimisation de la divergence dans les warps.
- Unrolling des boucles : Optimisation des performances avec des boucles déroulées.
- Utilisation des templates : Implémentation de fonctions génériques pour les réductions.

Tâche 3: Optimisation avancée et parallélisme dynamique

1. Implémentez une réduction parallèle avec exécution imbriquée en utilisant la parallélisation dynamique.
2. Comparez les performances de la réduction avec et sans exécution imbriquée.
3. Analysez la scalabilité du programme lorsque vous exposez plus de parallélisme.

Concepts couvertes:

- Parallélisme dynamique : Exposer plus de parallélisme en permettant aux threads de lancer de nouveaux threads.
- Exécution imbriquée : Implémenter des réductions parallèles imbriquées.

Tâche 4: Profiling et optimisation basée sur les profils

1. Utilisez `nvprof` pour optimiser le programme en identifiant les goulets d'étranglement et en mesurant les performances.
2. Maximisez l'occupation des warps et minimisez les latences en appliquant des techniques de masquage de la latence et de partitionnement des ressources.
3. Documentez les améliorations en termes de performances et d'efficacité.

Concepts couvertes:

- Optimisation basée sur les profils : Utilisation de `nvprof` pour améliorer les performances.
- Masquage de la latence et Partitionnement des ressources : Maximiser l'utilisation des ressources disponibles.
- Occupation des warps : Optimisation de l'occupation des ressources GPU pour une meilleure scalabilité.

4. Livrables

- Le code source pour chaque tâche avec des commentaires expliquant les optimisations effectuées.
- Un rapport d'analyse comprenant :
  - Les résultats des profils `nvprof` (avant et après optimisation).
  - Une explication des techniques d'optimisation appliquées et des résultats observés.
  - Une discussion sur les divergences des warps, l'optimisation de l'occupation des ressources, et la scalabilité.
  - Une comparaison des performances avec et sans exécution imbriquée dans la tâche 3.

5. Évaluation

Tâche 1: Introduction au modèle d'exécution CUDA et analyse avec `nvprof` (30 points)

- Correctitude du programme (10 points)
- Utilisation de `nvprof` pour profiler les warps et les opérations mémoire (10 points)
- Analyse des divergences et amélioration (10 points)

Tâche 2: Optimisation des performances et gestion de la divergence des warps (30 points)

- Réduction des divergences des warps (10 points)
- Implémentation correcte de l’unrolling (10 points)
- Utilisation des templates pour optimiser l'algorithme (10 points)

Tâche 3: Optimisation avancée et parallélisme dynamique (30 points)

- Implémentation de la parallélisation dynamique (10 points)
- Comparaison des performances avec et sans exécution imbriquée (10 points)
- Analyse de la scalabilité et exposition du parallélisme (10 points)

Tâche 4: Profiling et optimisation basée sur les profils (30 points)

- Utilisation de `nvprof` pour analyser les goulets d'étranglement (10 points)
- Maximisation de l’occupation et optimisation des ressources (10 points)
- Documenter les améliorations de performance (10 points)

Total possible: 120 points