Programmation et calcul générique sur GPU

Sommaire

1. Qu'est-ce que la "programmation GPU" ?

General-purpose computing on graphics processing units (GPGPU)

Usage du processeur graphique (*Graphical Processor Unit*, ou *GPU*)
pour effectuer des calculs typiquement effectués par
le processeur. Cela permet également d'effectuer des calculs
en parallèle, du fait de la structure du GPU et de la manière dont
les tratements sont effectués par le GPU.

TODO : il faudra entrer un peu plus en détails sur le fonctionnement d'une carte graphique (processeur, mémoire)

2. Petite histoire

Motivations : Calculs complexe sur un volume important de données + ou - complexes en parallelisant les tâches et en exploitant le processeur graphique (GPU).

Exemples: Multiplication matricielle et calcul sur des vecteurs en *N* dimensions.

Quelques expérimentations:

- 2001 : Multiplicaton matricielle exploitant le GPU
- 2005 : Décomposition LU (TODO kézako ?) sur GPU (plus rapide que sur CPU) - reférence nécessaire

Premiers applications de GPGPU utilisaient les shaders et les données ont
été encodées dans les textures. Un tel approche demande des bonnes connaissances
de OpenGL (avec GLSL) ou DirectX (avec HLSL).

Des bibliothèques spécifiques ont ensuite été développée pour répondre à cette problématique (CUDA, DirectCompute, OpenCL).

3. Les différentes bibliothèques

- DirectCompute (Microsoft)

  premièrement apparu dans DirectX 11, au contraire de HLSL DirectCompute est adapté aux calculs génériques.

- Compute Unified Data Architecture (CUDA) (Nvidia)

  permet de programmer en C, C++ et Fortran,
  mais pas trop portable (spécifique aux produits de Nvidia)

- OpenACC

  portable (a l'inverse de CUDA)

- Open Compute Library (OpenCL) (Khronos)

  originalement été créé pour programmer
  dans les systèmes hétérogènes (chaque unité peut
  être complètement différente des autres)

4. Open Computing Library (OpenCL)

4.1 Modèle des plateformes (Platform Model)

Modèle de OpenCL compose de host et des machines (au sens large)
connectées avec host (Compute Units). OpenCL-application s'exécute
sur host et envoie commands de bas niveaux aux unités de calculés.

4.2 Context d'exécution

A l'aide de API OpenCL host define context d'exécution qui contient:

- Noyaux (kernels): "fonctions" OpenCL.
- Les appareils (devices): ensemble de OpenCL-appareils utilisables par host.
- Objets de mémoire (Memory Objects): ensemble d'objets en mémoire,
  visibles par host et/ou devices.

4.3 Modèle de programmation (Programming Model)

Modèle d'execution de OpenCL utilise parallélisme des données (Data Parallel)
comme modèle principale de programmation.

On définit *calculs* comme une suite d'instructions appliquées à l'unite
de données exécutables en parallèle. À chaque instance du code (kernel),
qu'on appelle Work-Item, on associe un seul objet en mémoire.

Liens

- Calcul sur GPU - https://www.wikiwand.com/en/General-purpose_computing_on_graphics_processing_units#/Implementations