Introduction au GPU Computing : Architecture et Programmation

Mathias GUYON, Loïc TETREL

ÉTS
Le génie pour l'industrie

Résumé

Ce projet est une initiation à la programmation des cartes graphiques qui permettent d'effectuer de nombreux calculs en parallèle. La première partie de l'étude, concentrée sur l'architecture matériel, a permis d'appréhender les principes du parallélisme et de comprendre les avantages des GPU sur les CPU. Ensuite, l'apprentissage de la programmation CUDA et de la librairie NPP dans le cadre d'application simple a donné lieu à des études de performances démontrant la rapidité des GPU à traiter de grande quantité de données. Cette initiation débouchera sur une utilisation plus avancée dans le laboratoire LATIS de l'ETS.

Objectifs et méthodes

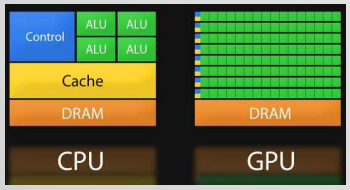
Pour arriver au bout de nos travaux, nous nous sommes fixés deux objectifs : (i) Nous initier à la programmation par GPU avec un apprentissage en ligne. (ii) Étudier la viabilité du sujet en effectuant une analyse de performance (multiplication matricielle et filtrage de Sobel horizontal).

Outils

Nous avons installés des outils liés à la programmation GPU (Visual Studio, OpenCV, CUDA et NPP) ainsi que divers outils d'analyse (HW Monitor, NSIGHT).

Visual Studio · NVIDIA CUDA · NVIDIA NPP · HWMonitor · NVIDIA Nsight · OpenCV

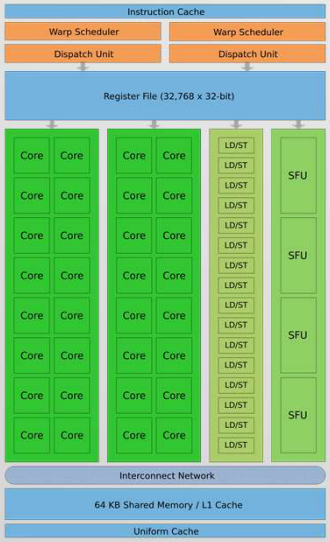
CPU et GPU



Architectures GPU

En 2006, NVIDIA introduit le GPU computing avec l'architecture TESLA. Ils proposent les cœurs CUDA avec le GPU « GeForce 8800 ». Pour le projet, nous disposons des cartes :

- GTX 560M : FERMI (2010)
- GTX 840M : MAXWELL (2014)



Un Streaming multiprocessor

Programmation GPU

Il existe trois façons de programmer mais nous en avons expérimentés deux.

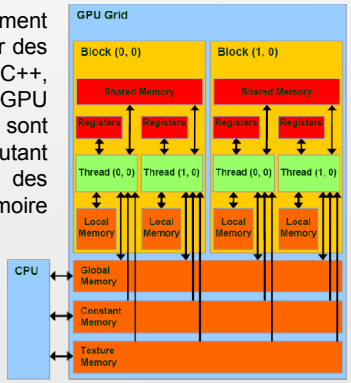
Librairies

L'utilisation de librairies permet d'effectuer des opérations courantes dans l'industrie (algèbre linéaire, transformation de Fourier…). Elles sont faciles d'utilisation et bénéficient de la puissance de nos cartes graphiques. Dans le cadre de notre projet, nous avons utilisé la librairie « CUDA NPP » proposant des milliers de fonctions de traitement d'images.

CUDA C/C++

CUDA est une architecture de traitement parallèle. Elle permet de développer des applications pour GPU en C, C++, Python… Les instructions pour le GPU sont codées dans des « kernel » et sont organisées en « threads » s'exécutant en parallèle. Les threads forment des « blocks » au sein desquels la mémoire est partagée.

Les blocs se répartissent en « grid », on attribue une grid à chaque kernel. Dans un kernel, on utilise les ID des threads et des blocs pour coder en parallèle.



Multiplication de matrices

La multiplication de matrice fait appel aux principes de base de la programmation CUDA et permet d'en expérimenter plusieurs aspects:

- Allocation de mémoire GPU
- Organisation des blocks et des threads
- Utilisation de la mémoire partagée

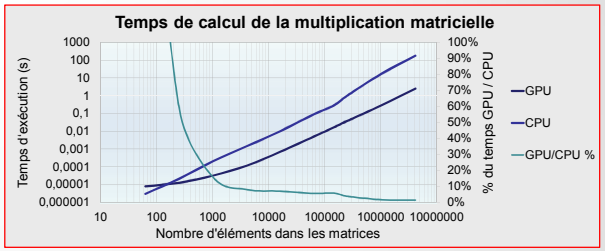
Les codes ci-dessous multiplient deux matrices de taille width*width stockées ligne par ligne dans deux tableaux A et B. Le résultat est stocké dans le tableau M.

Calcul séquentiel (CPU)

```
for (int i = 0; i < width; i++){
    for (int j = 0; j < width; j++){
        float sum = 0;
        for (int k = 0; k < width; k++){
            sum += A[i*width + k]
                 * B[k*width + j];
        }
        M[i*width + j] = sum;
    }
}
```

Calcul parallèle CUDA

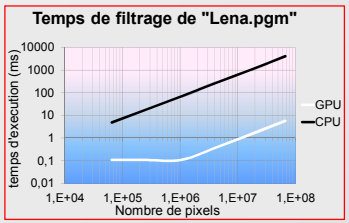
```
unsigned int row = blockIdx.y*blockDim.y + threadIdx.y;
unsigned int col = blockIdx.x*blockDim.x + threadIdx.x;

float Mvalue = 0;

for (int k = 0; k < width; k++)
      if (row < width && col < width)
            Mvalue += A[row*width + k]
                    * B[k*width + col];

M[row*width + col] = Mvalue;
```

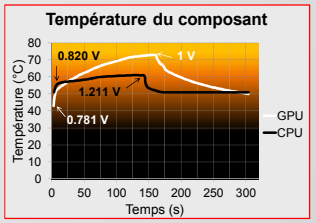


Filtrage de Sobel horizontal

L'opération a été effectuée sur des images codées sur 8 bits non signés. Nous avons effectuée l'analyse de performance en étudiant l'article de Lee et al [3]. La montée et descente du régime thermique a été étudié en arrêtant le programme au bout de 150s.

CPU : *OpenCV* GPU : Librairie *CUDA NPP*





Conclusion

Les GPU offrent des capacités de traitement rapide qui n'étaient pas exploitées dans l'industrie il y a quelques années. Aujourd'hui, Il existe des outils comme CUDA qui permettent une utilisation facile des ressources qu'elles offrent. L'accélération des applications est impressionnante et dans un futur proche, où la rapidité des CPU n'augmentera plus, la maîtrise du GPU computing sera un atout précieux. Le domaine de l'imagerie et de la vision par ordinateur semble particulièrement propice à cette technique puisqu'elle nécessite le traitement de grande quantité de données. C'est ce gain de temps qui permettra l'application temps réel de certains processus coûteux (SIFT, reconstruction 3D, imagerie satellitaire…).

Remerciements et références

Edgar GARCIA CANO

[1] https://developer.nvidia.com/cuda-zone

[2] https://www.youtube.com/user/NVIDIADeveloper

[3] Lee, V et al. (2010). *Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU.* ACM SIGARCH Computer Architecture News 38(3): 451-460.