ÉLÉMENT DE PORTFOLIO 02

Publication

1 DÉFINITION DE CET ÉLÉMENT

Titre de l'élément : Taming Voting Algorithms on GPUs for an Efficient Connected Component Analysis Algorithm [6]

URL de l'élément : `https://hal.archives-ouvertes.fr/hal-03330414`

2 MOTIVATIONS DU CHOIX DE CET ÉLÉMENT

Avec l'augmentation sans cesse croissante du nombre de cœurs des processeurs GPU, même la programmation *lock-free* à base d'instructions `atomic` est devenue inefficace. Cet article présente un algorithme de vote astucieux appliqué à l'étiquetage (ECC) et l'analyse en composantes connexes (ACC) sur GPU.

3 HISTORIQUE DES ALGORITHMES D'ÉTIQUETAGE ET D'ANALYSE EN COMPOSANTES CONNEXES

Les algorithmes d'ECC font partie des problèmes mal posés. Ils associent une étiquette unique à chaque groupe de pixels connexes d'une image binaire via un opérateur de voisinage qui détecte l'adjacence entre pixels. Les algorithmes modernes d'ECC et d'ACC dérivent tous de deux algorithmes pionniers : celui de Rosenfeld [8] qui est direct (en deux passes) mais qui a besoin d'une table d'équivalence et celui de Haralick [3] sans table d'équivalence, mais qui est itératif. La table d'équivalence est une structure Union-Find. Assigner une étiquette à une composante connexe revient donc à réaliser la fermeture transitive du graphe associé. Ce qui est rapide car la table d'équivalence possède en plus une relation d'ordre.

Si rapidement les algorithmes pour CPU se sont basés sur Rosenfeld, les premiers algorithmes pour GPU étaient basés sur Haralick car plus simples à mettre en œuvre. De plus la mémoire Shared permettait de faire des paquets d'itérations rapides. Mais rapide ne signifie pas efficace. Les principales étapes sont les suivantes :

- 1) Avoir l'idée de "plonger" la table d'équivalence dans l'image (aussi appelée notation linéaire) et 2) inventer un nouvel algorithme union-find *lock-free* (itération d'`atomic_min` jusqu'à a la stabilité). Cela fut co-inventé par Cabaret [2] et Komura [5].
- Playne et Hawick [7] ont inventé un algorithme énumérant les cas amenant à une équivalence entre étiquettes de ceux ne nécessitant que la propagation de l'étiquette courante. Cet algorithme générait beaucoup moins d'accès mémoire et était plus rapide. Les algorithmes d'ECC sur GPU venaient de changer de *paradigme* : il valait mieux faire beaucoup de tests – et donc de provoquer beaucoup de divergences de threads au sein d'un warp – que de faire des accès mémoire, même si ces cas sont rares et que les instructions atomic sont rapides.
- La seconde amélioration vient d'un doctorant de l'équipe [4] qui s'est inspiré de l'approche segment du LSL. Son algorithme HA manipule des sous-segments de la taille d'un warp (32). Tous les threads d'un warp déterminent leur position dans les sous-segments courants grâce à des intrinsics matériels. Et seul le premier thread de chaque sous-segment réalise un accès mémoire pour mettre à jour la table d'équivalence (ECC) ou pour voter (ACC). Le calcul des descripteurs de segment est aussi fortement accéléré par cette première approche segment.
- Alegretti et al. [1] ont aussi porté leurs algorithmes à base d'arbre de décision sur GPU. Et bien que l'arbre soit très grand, il est performant. Par contre il ne fait que de l'ECC et pas d'ACC.

La dernière évolution est le FLSL-GPU.

4 PRÉSENTATION DU FLSL-GPU

Lorsqu'on regarde l'évolution de la performance de ces différents algorithmes, on observe sur des images aléatoires que le pire cas empirique se situe au seuil de percolation (40% dans le cas 8-connexe) et que ce pire cas dégénère avec l'augmentation du nombre de coeurs. Il était à peine observable lors du passage d'une Jetson TX2 (256 CUDA cores) à une Jetson AGX (512 CUDA cores), mais était très problématique sur les grosses cartes massivement parallèles.

Le FLSL-GPU résout ce problème grâce à deux nouvelles avancées :

1. le traitement de segments complets (comme le LSL pour CPU) et,
2. la détection de conflits (CD), lorsque plusieurs threads veulent mettre à jour la même étiquette.

Ainsi, l'algorithme naif est en permanence inefficace avec 0.9 Gpixel/s. HA passe d'un débit de 4.22 Gpixel/s pour une granularité $g = 1$ à 25.8 Gpixel/s pour $g = 16$. Le FLSL+CD passe d'un débit de 24.5 à 170 Gpixel/s pour $g = 16$. LE FLSL-GPU est ainsi de $\times 4$ à $\times 10$ plus rapide que HA en ACC. A notre connaissance, il n'y a pas d'algorithme d'ACC sur GPU à part ceux de l'équipe.

5 RÉFÉRENCES BIBLIOGRAPHIQUES

[1] Stefano Allegretti, Federico Bolelli, and Costantino Grana. Optimized Block-Based Algorithms to Label Connected Components on GPUs. *IEEE Transactions on Parallel and Distributed Systems*, 2019.

[2] L. Cabaret, L. Lacassagne, and D. Etiemble. Distanceless label propagation : an efficient direct connected component labeling algorithm for GPUs. In *International GPU Technical Conference (GTC)*, 2017.

[3] R.M. Haralick. Some neighborhood operations. In *Real-Time Parallel Computing Image Analysis*, pages 11–35. Plenum Press, 1981.

[4] A. Hennequin and L. Lacassagne. A new direct connected component labeling and analysis algorithm for GPUs. In *GPU Technology Conference (GTC)*, 2019.

[5] Y. Komura. Gpu-based cluster-labeling algorithm without the use of conventional iteration : application to swendsen-wang multi-cluster spin flip algorithm. *Computer Physics Communications*, pages 54–58, 2015.

[6] F. Lemaitre, A. Hennequin, and L. Lacassagne. Taming voting algorithms on GPUs for an efficient Connected Component Analysis Algorithm. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[7] D. P. Playne and K. Hawick. A new algorithm for parallel connected-component labelling on GPUs. *IEEE Transactions on Parallel and Distributed Systems*, 2018.

[8] A. Rosenfeld and J.L. Platz. Sequential operator in digital pictures processing. *Journal of ACM*, 13,4 :471–494, 1966.