GPU Programming

Tercera Entrega

Diego Madariaga Román

Resumen

La programación en GPU puede definirse como el uso de una unidad de procesamiento gráfico (GPU) en combinación con una CPU para acelerar aplicaciones de distintos tipos. Este tipo de programación ha ido tomando gran importancia dentro del desarrollo de software durante los últimos años, y actualmente, muchos algoritmos comúnmente utilizados en diferentes áreas de la computación, como procesamiento de imágenes, inteligencia artificial, teoría de grafos y minería de datos, han sido modificados para poder experimentar su ejecución en GPU.

Es impresionante la cantidad de métodos computacionales con los que se ha experimentado la computación acelerada por GPU, por lo que es difícil encontrar algún algoritmo frecuentemente utilizado y con potencial para ser ejecutado en una GPU, con el cual aún no se haya investigado al respecto.

Una de las áreas fuertemente explotadas por la programación en GPU, es la del procesamiento de imágenes, en donde toman gran importancia las técnicas correspondientes a Imagen Médica e Imagen Biológica, siendo una de las herramientas más utilizadas en dichas áreas la segmentación de imágenes, la cual suele ser realizada utilizando métodos de clustering.

En el proyecto a llevar a cabo se implementará, utilizando la plataforma CUDA para codificar algoritmos en GPU de NVIDIA, un algoritmo de segmentación de imágenes basado en la técnica de clustering K-Means, y se realizarán experimentos de comparación con su versión ejecutable en CPU, con el fin de determinar la conveniencia o no del uso de programación en GPU en este caso.

El trabajo a realizar, comprenderá un estudio preliminar de algunos métodos de segmentación de imágenes ya implementados y que utilicen la unidad de procesamiento gráfico [1] y especialmente desarrollados para la plataforma CUDA [6]. Además, se estudiará la implementación para programación en GPU de los métodos de clustering (para uso general) K-Means [2] [7] y Mean Shift [5], con los cuales se espera desarrollar en la plataforma CUDA, una versión en GPU de segmentación de imágenes que haga uso de dichos métodos.

1. Situación Actual

1.1. Arquitectura GPU

Con el fin de comprender las características de la unidad de procesamiento gráfico que permiten obtener ventajas frente al uso de CPUs para algunos problemas, es importante conocer su modelo de arquitectura. La arquitectura de una GPU, consta de varios Multiprocesadores de Streaming (SM), los cuales contienen *cores* de ejecución que comparten una memoria caché L1, una memoria compartida para el traspaso de datos entre *cores* y una unidad de búsqueda de instrucciones (Figura 1).

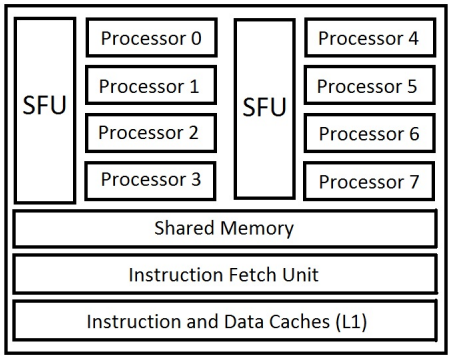


Figura 1: Multiprocesador de Streaming y sus componentes

La principal tarea de un SM es coordinar la ejecución en paralelo, ya que dentro de un SM solamente pueden haber bloques de 32 threads, llamados *warps*, ejecutándose de forma concurrente. Una GPU completa, cuenta con varios SM, los que comparten una memoria caché L2 y una memoria DRAM (Figura 2).

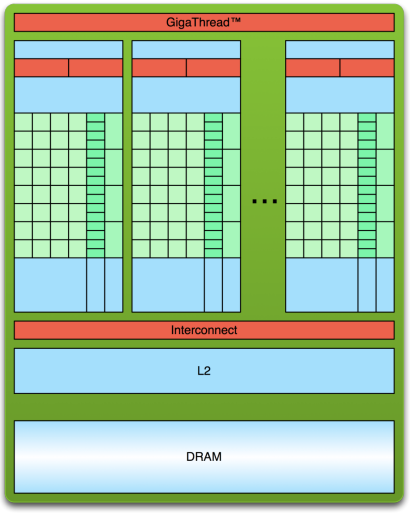


Figura 2: Arquitectura GPU completa

La principal diferencia entre CPUs y GPUs, radica en que estas últimas comprenden hardware especializado para realizar computación intensiva y con alto grado de paralelización, exactamente las características de los algoritmos de renderizado gráfico. Más específicamente, las GPUs se adecúan de buena forma a algoritmos en donde una misma secuencia de instrucciones, con alta computación aritmética, se aplica a distintos elementos de forma paralela. Esto resulta muy eficiente dado que en cada SM se procesa una misma instrucción en todos los threads del *warp* en ejecución al mismo tiempo.

1.2. Programación GPU para segmentación de imágenes

Para los procesos de segmentación de imágenes, en el ámbito de la imagenología o imagen médica, en donde dichos procesos son fuertemente requeridos, se ha incursionado en el uso de programación GPU para optimizar los algoritmos comúnmente utilizados [6]. Las técnicas de segmentación implementadas que han tenido buenos resultados experimentales al utilizar imágenes médicas son:

- *Region Growing*, el cual es un procedimiento iterativo que a partir de puntos iniciales crea regiones, las cuales en cada iteración se van expandiendo hacia sus vecinos (en caso de que se cumplan ciertas condiciones).
- *Watershed*, que analiza la imagen como un relieve topográfico, en donde el nivel del color de cada píxel representa la altura del plano en ese punto. Posteriormente, se ejecuta un "llenado de agua", en donde el agua corre hacia los sectores de baja altura, agrupándose y formando una segmento independiente.

Ambas técnicas mencionadas han sido implementadas en la plataforma CUDA, y a pesar de que según los propios autores de estas implementaciones, no se obtuvieron resultados plenamente satisfactorios, se logró mejorar el tiempo de ejecución de los algoritmos, en comparación con sus símiles de ejecución exclusiva en CPU.

Otro método de segmentación de imágenes con el cual se ha trabajado la computación acelerada por GPU, es el método de *Quick Shift* [1], el cual es un algoritmo de búsqueda de modo rápido o *fast mode seeking algorithm*, que segmenta una imagen de al menos 2 canales, por medio de la identificación de clusters de píxeles en las dimensiones espaciales y de colores, utilizando una segmentación de tipo jerárquica. La implementación de este algoritmo, también desarrollada en CUDA, presentó muy buenos resultados experimentales en términos, principalmente, del *speedup* obtenido en relación a la implementación del algoritmo en CPU, aunque no se muestra explícito el algoritmo utilizado en CPU.

Aparte de los descritos anteriormente, no se encontró información que diera cuentas de otro algoritmo de segmentación de imágenes con el que se haya experimentado su ejecución en GPU. Sin embargo, otras técnicas tales como *Histogram-Based Clustering*, *Graph-Based Segmentation*, *K-Means* y *Mean Shift*, las cuales pueden ser utilizadas para segmentación de imágenes, ya han obtenido versiones eficientes en ambientes *multi-thread* [3] [8] en CPU, lo que evidencia de que pueden ser tratados eficientemente de forma paralela. Además, los algoritmos de *K-Means* [2] [7] y *Mean Shift* [5] han sido puestos bajo el enfoque de la programación GPU, aunque con un enfoque no puesto en la segmentación de imágenes, sino que en procesos de clustering generales y en algoritmos de tracking de videos [5], en donde se han obtenido buenos resultados, aunque nuevamente, sin hacer alusión explícita de la naturaleza de los algoritmos CPU con los cuales se han comparado.

Finalmente, tomando como punto de partida los resultados obtenidos en los trabajos anteriormente mencionados, se implementará un algoritmo de segmentación de imágenes haciendo uso de la programación GPU, mediante la plataforma CUDA y de la siguiente técnica de clustering:

- *K-Means*, método iterativo que encuentra las medias de las distribuciones de los píxeles (o clusters), para lo cual es necesario saber en un principio el número K de clusters en los cuales el algoritmo deberá segmentar la imagen.

Dicho algoritmo posee características que lo posicionan como un buen algoritmo con el cual experimentar la programación GPU, ya que es un método iterativo en donde en cada iteración se realizan las mismas instrucciones para cada uno de los píxeles de la imagen, las cuales al ser independientes para cada píxel, pueden ser ejecutadas en paralelo. Además, dichas instrucciones requieren de un alto grado de computación aritmética, dado que se requiere calcular para cada píxel la distancia euclidiana con cada uno de los k centroides. Para dar mayor objetividad a los resultados obtenidos, se realizará una comparación con el mismo algoritmo pero ejecutado en paralelo en una CPU.

2. Experimentación

Un error muy usual en las investigaciones de algoritmos implementados en GPU, es que al momento de hacer el análisis de los resultados, se suele realizar una comparación con la implementación secuencial del algoritmo ejecutado en CPU. Esta práctica suele sobrestimar los resultados obtenidos, ya que comúnmente se obtienen valores de *speedup* del orden de 50X o incluso 100X. Una comparación GPU-CPU más justa [4], sugiere la comparación de los algoritmos ejecutados en GPU con la versión optimizada del algoritmo multithread ejecutado en CPU, lo cual además, se suele hacer comparando las ejecuciones en una GPU y una CPU pertenecientes a una misma máquina.

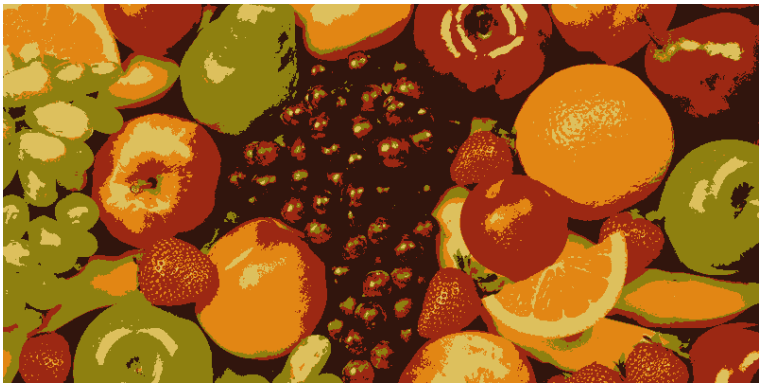
Dado lo anterior, se realizó la implementación del algoritmo de segmentación de imágenes utilizando la técnica de K-Means en la extensión de C/C++ de la plataforma CUDA, para su ejecución en GPU, y se utilizó la extensión de C/C++ de *CilkPlus* para implementar el algoritmo multithread para su ejecución en CPU. Las especifícaciones de las arquitecturas de GPU y CPU utilizadas son las siguientes:

- GPU: NVidia GeForce GTX 750 Ti
  - 5 Multiprocesadores de Streaming
  - 128 *cores* por procesador
  - Velocidad del reloj (GPU): 1,14 GHz
  - Velocidad del reloj (Memoria): 2,7 GHz
  - Tamaño del bus de Memoria: 128 bits
- CPU: Intel(R) Core(TM) i5-3570
  - Arquitectura: x86_64
  - 4 *cores*
  - Velocidad del reloj (CPU): 1,6 GHz
  - Tamaño del bus de Memoria: 64 bits

El código de la implementación se encuentra disponible en un repositorio *GitHub*[1], donde se encuentran además, instrucciones para la ejecución de los programas y la ejecución de experimentos. Ambas implementaciones toman como input una imagen en formato PPM, el número k de clusters a encontrar, y la cantidad de iteraciones a realizar; y entrega la imagen resultante del proceso de segmentación en un archivo también de formato PPM. Un ejemplo del correcto funcionamiento de los algoritmos se puede observar en la Figura 3.

[1] `https://github.com/dmadariaga/gpu-programming-research`

(a) Imagen original



(b) Imagen segmentada

Figura 3: Ejemplo de segmentación de imagen con k=5 y número de iteraciones = 1000

Para la realización de los experimentos, se ejecutaron reiteradas veces los algoritmos en GPU y en CPU (utilizando los 4 cores disponibles). En estas ejecuciones, se variaron los valores del número k de clusters, el número de iteraciones, y el tamaño de la imagen de entrada. A continuación, se muestra el resumen de los resultados obtenidos en las ejecuciones sobre una imagen de 1600 píxeles.

Tabla 1: Tiempo de ejecución [s] de algoritmo K-Means en GPU y CPU para una imágen de 1600 píxeles

k	**Ejecutado en**	**Número de iteraciones**		
		100	1000	5000
5	GPU	0,001129	0,72429	4,609269
	CPU	0,955862	9,551453	48,039376
10	GPU	0,001127	1,058918	6,580127
	CPU	1,842428	18,786756	92,237162
20	GPU	0,00113	1,770733	11,132368
	CPU	3,621773	36,132279	181,738398

De los resultados obtenidos, es posible ver una clara tendencia de la ejecución en GPU de ser más rápida que la ejecución en CPU, en donde para valores grandes en el número de iteraciones,

obtenemos valores de *speedup* cercanos a 15X. Esto sustenta el planteamiento establecido de que el algoritmo de segmentación utilizando K-Means presentaba ventajas intrínsecas que le permitirían tener un buen desempeño al ser ejecutado en una GPU.

3. Conclusiones

A modo de conclusión, es posible mencionar que se logró implementar correctamente y utilizando la plataforma CUDA, un algoritmo de segmentación de imágenes basado en el método de K-Means. En el algoritmo se podían observar las dos principales caracteristicas que todo buen algoritmo en GPU debe explotar: tareas en paralelo que ejecutan las mismas instrucciones y tareas computacionalmente costosas, las cuales permitieron que se lograra un buen desempeño al ser ejecutado en una GPU, ya que saca provecho de las características de éste hardware.

Los resultados obtenidos verifican a su vez el buen comportamiento que tiene el uso de GPUs para el área de procesamiento de imágenes, ya que los algoritmos utilizados suelen presentar independencia entre el trabajo a realizar sobre cada píxel de la imagen.

Finalmente, a pesar de que los resultados experimentales fueron favorables para el algoritmo implementado en GPU, es necesario observar que el *speedup* estimado con respecto a la ejecución del algoritmo en CPU podría estar sobrestimado dadas las siguientes posibilidades:

- Los algoritmos pueden no estar optimizados, tanto como para la ejecución en GPU como en CPU, ya que su implementación consideró un proceso fundamentalmente pedagógico.
- Dentro de la máquina donde se realizaron los experimentos, la tarjeta gráfica fue incorporada hace algún tiempo justamente por sus buenas especificaciones, por lo que la comparación entre GPU y CPU, podría tomar mayor sentido si se utilizara una CPU también con mejores carácterísticas.

Para solucionar los últimos puntos, queda como tarea futura realizar un estudio profundo para comprender cuál es realmente una buena forma de realizar comparaciones GPU-CPU, lo cual es un problema muy presente en las publicaciones actuales relacionadas al tema. Además, se propone un estudio más detallado de la arquitectura GPU, con el fin de poder realizar optimizaciones a los algoritmos tomando como base los procesos físicos que ocurren internamente, como el uso de memorias caché y memorias compartidas.

Referencias

[1] Fulkerson B., Soatto S. Really quick shift: Image segmentation on a GPU. *Workshop on Computer Vision using GPUs, held with the European Conference on Computer Vision*, 2010.

[2] Hong-Tao B., Li-li H., Dan-tong O., Zhan-shan L., He L. K-Means on Commodity GPUs with CUDA. *2009 WRI World Congress on Computer Science and Information Engineering LosAngeles*, 2009.

[3] Khotanzad A., Bouarfa A., 1990, Image segmentation by a parallel, non-parametric histogram based clustering algorithm. *Pattern Recognition, Volume 23, Pages 961-973*, 1990.

[4] Lee V, Kim C, Chhugani J, Deisher M, Kim D, Nguyen A, et al. Debunking the 100x gpu vs. cpu myth: an evaluation of throughput computing on cpu and gpu. *Proceedings of the 37th Annual International Symposium on Computer Architecture: 19-23, France*, 2010.

[5] Li, P., Xiao, L. Mean shift parallel tracking on GPU. *Iberian Conference on Pattern Recognition and Image Analysis, pp. 120–127*, 2009.

[6] Pan L., Gu L., Xu J. Implementation of medical image segmentation in CUDA. *International Conference on Information Technology and Applications in Biomedicine (ITAB), pp. 82–85.*, 2008.

[7] Shalom S.A.A., Dash M., Tue M. Efficient K-Means Clustering Using Accelerated Graphics Processors. *2009 WRI World Congress on Computer Science and Information Engineering LosAngeles*, 2008.

[8] Wassenberg J., Middelmann W., Sanders P. An efficient parallel algorithm for graph-based image segmentation. *CAIP '09: Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns, pages 1003–1010, Berlin,* 2009.