Aceleración de un Programa de Reconstrucción Iterativa para PET mediante el uso de GPUs

J. López Herraiz[1], S. España Palomares[2], E. Herranz Muelas[1], M. Desco Menéndez[3,4], J. J. Vaquero López[3], J. M. Udias Moinelo[1]

[1] Grupo de Física Nuclear, Dpto. Física Atómica, Molecular y Nuclear, Universidad Complutense de Madrid.

[2] Department of Radiation Oncology, Massachusetts General Hospital and Harvard Medical School, Boston, EEUU.

[3] Departamento de Bioingeniería e Ingeniería Aeroespacial, Universidad Carlos III de Madrid.

[4] Unidad de Medicina y Cirugía Experimental, Hospital General Universitario Gregorio Marañón, Madrid.

email: joaquin@nuclear.fis.ucm.es

Resumen

En este trabajo se muestran los principales pasos seguidos para modificar el programa de reconstrucción de imagen tomográfica FIRST (Fast Iterative Reconstruction Software for (PET) Tomography) de forma que las partes de la reconstrucción que requieren un mayor tiempo de cálculo (proyección y retroproyección) puedan ser ejecutadas en las unidades de procesamiento gráfico (GPUs) de un ordenador. En la actualidad, las GPUs superan a las CPUs en capacidad de cómputo en paralelo,. El código original ha sido reescrito en C con extensiones de CUDA, buscando en todo momento hacer un uso óptimo de las propiedades de la GPU. De esta forma se ha logrado reducir el tiempo requerido para realizar cada reconstrucción en un factor mayor de 70. Esta mejora tan notable en el tiempo de reconstrucción abre la puerta a un uso más generalizado de este tipo de métodos iterativos.

1. Introducción

La reconstrucción de imagen en tomografía por emisión de positrones (PET) es muy costosa desde el punto de vista computacional, especialmente cuando se emplean métodos iterativos basados en modelos realistas de la emisión y detección de la radiación [1]. Los modernos procesadores *multi-core* y la mejora de los programas de reconstrucción como FIRST [1] han logrado reducir significativamente el tiempo requerido para realizar una reconstrucción iterativa completa de una adquisición, hasta llegar a hacerla compatible con su uso clínico o preclínico ordinario.

Sin embargo, el crecimiento en complejidad de los modernos escáneres PET, así como el uso cada vez más común de protocolos avanzados de adquisición como pueden ser los estudios dinámicos, hacen que el número de datos a reconstruir y la demanda computacional sea cada vez mayor [2]. Esto ha llevado a plantear el uso de *hardware* alternativo a los procesadores convencionales (CPU) para realizar los cálculos requeridos en la reconstrucción. Una de las alternativas más prometedoras la constituyen las tarjetas gráficas (GPUs), que, a pesar de que inicialmente tuvieron un uso dedicado al mundo de los videojuegos, han ido ampliando sus campos de aplicación, hasta llegar a la actualidad, en la que las tarjetas gráficas de propósito general (GP-GPU) se emplean en multitud de proyectos científicos y tecnológicos [3]. Las GPUs pueden manejar grandes cantidades de datos en paralelo trabajando en modo SIMD (una instrucción, múltiples datos), superando de esta forma a los procesadores en capacidad de cómputo. La reconstrucción tomográfica puede verse especialmente favorecida por esta paralelización masiva dado que las dos partes del código que consumen un mayor tiempo de cálculo (proyección y retroproyección) pueden ser organizadas fácilmente en modo SIMD y distribuidas en las unidades de cálculo disponibles, asignando una parte de los datos a cada unidad [4].

La principal dificultad que planteaba el uso de las GPUs para el cálculo científico consistía en que para programarlas hacía falta un conocimiento profundo de sus características, al tener que implementar los algoritmos en términos específicos de las GPUs como son los vértices y las texturas [5]. Los recientes avances en la facilidad de programación de las GPUs han solventado en gran medida este problema. CUDA [3], desarrollado por NVIDIA, ofrece la posibilidad de crear programas en lenguaje C estándar junto con una serie de extensiones para hacer uso de la GPU. Programada a través de CUDA, la GPU se puede ver como un dispositivo capaz de ejecutar en paralelo un gran número de cálculos.

En los últimos años se han creado diversos programas de reconstrucción tomográfica para CT y PET basados en el uso de la GPU [2], [5-6]. Sin embargo, en general, se tratan de programas que usan aproximaciones y métodos distintos y simplificados respecto a los que se emplearían con la CPU.

En este trabajo se ha buscado realizar una implementación en CUDA del programa de reconstrucción iterativa FIRST para PET que fuese lo más fiel posible al código original. Nuestro principal objetivo ha sido obtener una aceleración significativa del código sin que la calidad de las imágenes se viese comprometida. Además, se ha buscado generar un código sencillo y flexible que permita futuras modificaciones y adaptaciones sin un excesivo esfuerzo adicional.

El programa ha sido desarrollado sin tener en cuenta en general el modelo específico de GPU de NVIDIA que

ejecutará el código. Sin embargo, ha sido necesario prestar una especial atención a los usos de la memoria de la GPU con el fin de alcanzar tiempos de reconstrucción óptimos.

2. Materiales y métodos

2.1. Descripción general del código para CPU

Para una descripción más detallada del código de reconstrucción FIRST, el lector puede consultar la referencia [1]. En esta sección nos limitaremos a describir sus principales componentes y características.

El programa FIRST implementa un código de reconstrucción iterativa 3D-OSEM [1] basado en un modelo realista de la emisión y detección de la radiación. Este modelo (SRM) se genera mediante el código Monte Carlo PeneloPET [7] basado en PENELOPE [8]. La SRM consiste en el conjunto de probabilidades C_{ij} de que una desintegración producida en un determinado voxel j de la imagen genere una coincidencia en un determinado par de cristales del detector (que constituyen una línea de respuesta (LOR) i).

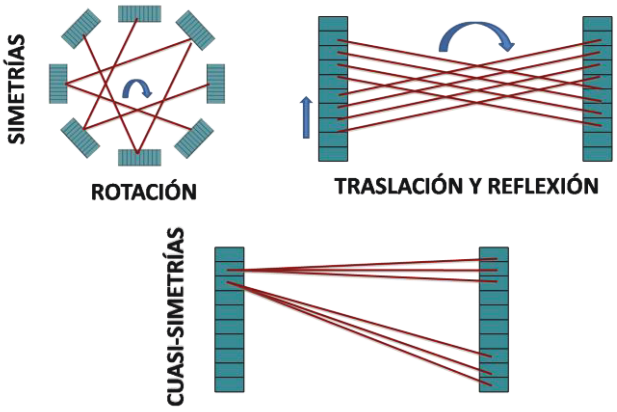

Figura 1. *Simetrías y Cuasi-simetrías en un escáner PET. Todas estas LORs tienen una distribución de probabilidad similar.*

Debido a su elevado tamaño, es necesario hacer uso de todas las simetrías presentes en el sistema con el fin de que la SRM pueda ser almacenada en la memoria RAM del ordenador durante la ejecución del programa. Esto es importante para alcanzar tiempos de ejecución óptimos. Empleando las simetrías del sistema, sólo algunas distribuciones de probabilidad correspondientes a ciertas LORs requieren ser simuladas y almacenadas, dado que LORs simétricamente equivalentes (Figura 1) tendrán la misma distribución de probabilidad. Sin embargo, en muchos casos, el uso de las simetrías no es suficiente. En el programa FIRST se hizo uso de lo que se denominó cuasi-simetrías [1]: LORs con un ángulo muy similar respecto a los cristales tienen una distribución de probabilidad muy similar (Figura 1). Por tanto, usando las simetrías y las cuasi-simetrías, sólo hace falta simular y almacenar ciertos LORs elegidos, denominados Super-LORs, sin que la calidad de las imágenes reconstruidas se vea afectada.

El algoritmo de reconstrucción empleado es el 3D-OSEM [9]. En él, la imagen reconstruida X_j se va multiplicando en cada iteración por una imagen de correcciones obtenida como la media ponderada de un subconjunto S de factores de corrección (1).

Estos factores se obtienen como el cociente entre los datos medidos Y_i y los datos estimados P_i a partir de la imagen y los términos C_{ij} que constituyen la SRM.

$$X_j^{(t+1)} = X_j^{(t)} \cdot \frac{\sum_{i \in S} C_{ij} \cdot \left[Y_i \,/\, P_i \right]}{\sum_{i \in S} C_{ij}} \qquad (1)$$

$$P_i = \sum_j C_{ij} \cdot X_j^{(t)}$$

La Figura 2 muestra esquemáticamente los principales pasos que se realizan en cada iteración del algoritmo.

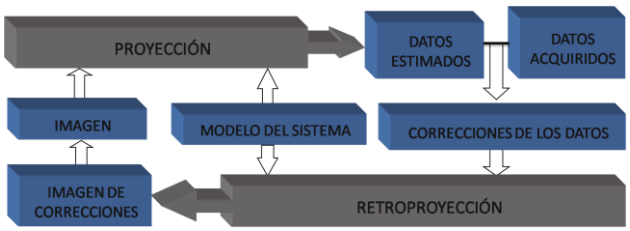

Figura 2. *Esquema del código iterativo implementado en el programa FIRST-CPU.*

En este trabajo, se presenta una versión del código adaptado para un escáner PET de pequeños animales compuesto por un par de detectores en rotación en coincidencia. El número de vóxeles en la imagen reconstruida fue de 175 × 175 × 59 y el número de datos en el sinograma fue de 175 (bines radiales) × 130 (ángulos) × (30 × 30) (sinogramas). Los datos usados en este estudio corresponden a una simulación obtenida con PeneloPET de un maniquí tipo Derenzo con cilindros de distinto diámetro rellenos de ^{18}F. El código inicial escrito en FORTRAN fue reescrito en lenguaje C y se compiló usando las mejores opciones de optimización. Para obtener los tiempos de reconstrucción y una comparación más sencilla con la GPU se empleó una única CPU.

2.2. Descripción general del código para GPU

Con el gran número de cálculos que se pueden realizar en paralelo en la GPU, la principal limitación en este tipo de implementaciones es el acceso a memoria. En la Tabla 1 se muestran los distintos tipos de memoria disponibles en la GPU a través de CUDA. Se observa que los accesos a texturas son mucho más rápidos que a la memoria global. Así mismo, hay una pequeña cantidad de memoria de acceso rápido correspondiente a registros, constantes y memoria compartida cuyo uso apropiado permite también agilizar los tiempos de ejecución del código.

Tipos de memoria de la GPU	Número de Ciclos de Reloj

Registro	1
Mem. compartida	1
Constantes	1-10 (caché), 10-100 (no caché)
Texturas	1-10 (caché), 10-100 (no caché)
Global	400-600

Tabla 1. *Ciclos de reloj requeridos para acceder a los distintos tipos de memoria en la GPU.*

Por tanto, para reducir los tiempos de acceso a memoria, en este trabajo se buscó explotar las texturas 3D disponibles en CUDA. Al iniciarse el programa, la SRM se carga en la GPU como textura 3D y permanece allí inalterada durante toda la ejecución. Es importante resaltar que esto es posible debido a que el uso de las simetrías y cuasi-simetrías descrito en el apartado anterior logra reducir el tamaño de la SRM para que quepa en la memoria reservada a texturas de la GPU. Además de esta textura, se define otra textura 3D correspondiente a la imagen que se va reconstruyendo, así como otra más para las correcciones a aplicar (Figura 3).

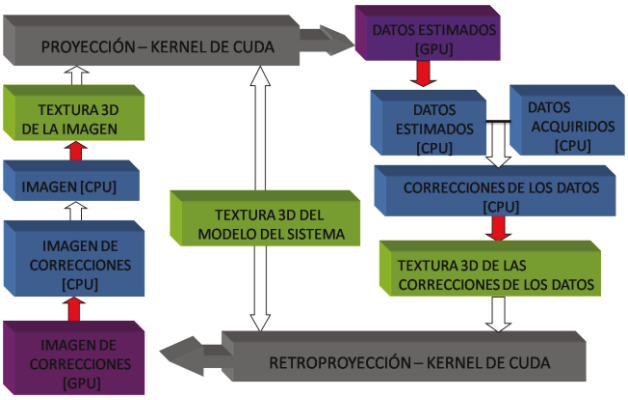

Figura 3. *Esquema del código iterativo implementado en el programa FIRST-GPU.*

La Figura 3 muestra los principales pasos realizados en cada iteración en el código FIRST implementado para GPU. Se observa que existen una serie de transferencias de datos e imágenes entre la CPU y la GPU (indicados en rojo). Sin embargo, debido al reducido tamaño de las imágenes PET, estas transferencias no suponen un coste de tiempo adicional muy significativo. Las partes que requieren un mayor tiempo de ejecución (proyección y retroproyección) se ejecutan en la GPU, mientras que el resto de operaciones se realizan en la CPU, como en el código original. De esta forma, la parte del código específica de CUDA se ve reducida al máximo.

2.3. Comparativa de tiempos de reconstrucción

Una vez implementado el código de reconstrucción en CUDA, se realizó una comparativa de los tiempos de ejecución usando la CPU y la GPU. Las CPUs empleadas son de las más rápidas que existen en la actualidad en el mercado de PCs. Como ya se ha indicado, los tiempos corresponden a la ejecución sobre una única CPU. Por otro lado, el código en GPU se ejecutó en dos tarjetas gráficas muy distintas. La primera de ellas (GT 120) es una tarjeta muy básica y económica, con sólo 2 multiprocesadores (SM), mientras que la otra (TESLA C60) es una de las más potentes que existen en la actualidad y dispone de 27 SM.

3. Resultados

La Tabla 2 muestra los tiempos de reconstrucción empleados en las distintas arquitecturas para reconstruir la adquisición simulada. Las imágenes reconstruidas en CPU y GPU son iguales como se observa en la Figura 4.

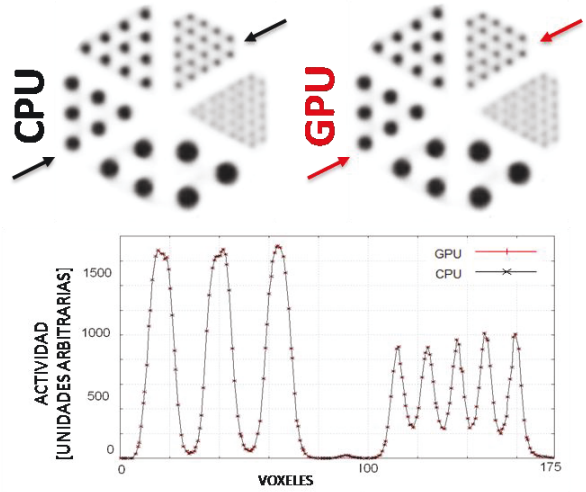

Figura 4. *Arriba: Imágenes reconstruidas en la CPU y en la GPU a partir de la misma adquisición. Abajo: Perfil a lo largo de la imagen.*

Arquitectura	Tiempo (s)	Factor de Aceleración
CPU - Intel(R) Xeon(R) CPU X5472 @ 3.00GHz	2048	-
CPU - Intel(R) Core(TM) i7 CPU 940 @ 2.93GHz	1561	1x
GPU - GT 120 1.0 GB – 2 SM	199	8x
GPU - TESLA C60 2.0 GB - 27 SM	22s	72x

Tabla 2. *Tiempos de reconstrucción de la adquisición del maniquí Derenzo en distintas arquitecturas.*

4. Conclusiones

Se ha implementado con éxito una versión del programa FIRST de reconstrucción iterativa 3D para adquisiciones PET en CUDA, para poder ser ejecutado en la GPU. Se ha logrado de esta manera una mejora muy significativa del tiempo de reconstrucción. Este resultado es más destacado considerando que FIRST ya implementaba un código muy optimizado. La calidad de la imagen obtenida en la GPU no difiere de la que se obtiene con la CPU.

Se puede apreciar de los resultados de la Tabla 2, que la mejora en los tiempos alcanzada depende en gran medida del tipo de GPU empleada. Esto es de interés, ya que permitirá hacer un uso ventajoso de las futuras generaciones de GPUs que dispondrán de un número mayor de multiprocesadores.

Agradecimientos

Este trabajo ha sido financiado en parte por el MEC (FPA2007-62216), la UCM (Grupos UCM, 910059), el CPAN (Consolider-Ingenio 2010, CSPD-2007-00042), la red RECAVA-RETIC, el proyecto ARTEMIS (S2009/DPI-1802, Fondo Europeo de Desarrollo Regional) y el proyecto ENTEPRASE (PSE-300000-2009-5, MICINN. España).

Referencias

[1] Herraiz JL, España S, Vaquero JJ, Desco M, Udias JM, FIRST: Fast Iterative Reconstruction Software for (PET) tomography. *Phys. Med. Biol*., vol. 51, 2006, pp. 4547, (ISSN: 0031-9155).

[2] Pratx G, Chinn G, Olcott PD and Levin CS, Fast, accurate and shift-varying line projections for iterative reconstruction using the GPU. *IEEE Trans. Med. Imaging*, vol. 28, no. 3, 2009, pp. 435-445, (ISSN: 0278-0062).

[3] Página web de NVIDIA-CUDA. http://www.nvidia.es/object/cuda_apps_flash_new_es.html (Consultada: Agosto 2010)

[4] Hong IK et al. Fast Symmetry and SIMD-Based Projection-Backprojection (SSP) Algorithm for 3-D PET Image Reconstruction. *IEEE Trans. Med. Imaging*, vol. 26, no. 6, 2007, pp 789-803, (ISSN: 0278-0062).

[5] Xu F and Mueller K .Real-time 3D computed tomographic reconstruction using commodity graphics hardware. *Phys. Med. Bol*., vol. 51, 2007, pp. 3405-3419, (ISSN: 0031-9155).

[6] Keck B et al.. High Resolution Iterative CT Reconstruction using Graphics Hardware. *Nucl. Sci. Symp. and Med. Imag. Conf. IEEE*, 2009, pp. 4035 - 4040. (ISSN: 1082-3654).

[7] España, S. et al., "PeneloPET, a Monte Carlo PET simulation toolkit based on PENELOPE: Features and validation," *Phys. Med. Biol*., vol. 54, no. 6, pp. 1723–1742, 2009 (ISSN: 0031-9155).

[8] Baró, J, et al., "PENELOPE: an algorithm for Monte Carlo simulation of the penetration and energy loss of electrons and positrons in matter," *Nucl. Inst. Meth. In Phy. Res. B*, vol. 100, pp. 31-46, 1995 (ISSN: 0168-583X)

[9] Hudson, HM and Larkin, RS. Accelerated image reconstruction using ordered subsets of projection data. *IEEE Trans. Med. Imaging,* vol. 13, 1994, pp. 601–609, (ISSN: 0278-0062).