Paralelismo y Distribución para la Optimización del Proceso de Visualización de Volúmenes en Tiempo Real

C. Perez-Monte, F. Piccoli
LIDIC- Universidad Nacional de San Luis
Ejército de los Andes 950
Tel: 02652 420823, San Luis, Argentina
{*mpiccoli*}*@unsl.edu.ar*
{*cristian.perez*}*@gridtics.frm.utn.edu.ar*

1. Contexto

Esta propuesta de trabajo se lleva a cabo dentro de la línea de Investigación "Computación de Alto Desempeño" del proyecto "Nuevas Tecnologías para un tratamiento integral de Datos Multimedia". Este proyecto es desarrollado en el ámbito del Laboratorio de Investigación y Desarrollo en Inteligengia Computacional (LIDIC) de la Universidad Nacional de San Luis.

2. Resumen

Los métodos actuales de simulación, entrenamiento y planeamiento de procedimientos quirúrgicos basados en la tecnología de realidad virtual utilizan datos volumétricos producidos por escáneres de resonancia magnética (MRI) y tomografía axial computada (CT) de pacientes reales. Considerando el estado actual de la tecnología de renderizado de imágenes médicas, la visualización en tiempo real de los grandes volúmenes de información extraída de estos dispositivos presenta un desafío: desarrollar nuevos métodos de procesamiento donde se respeten las severas exigencias de resolución y calidad, además de permitir una gran fluidez en la animación para obtener un nivel suficientemente alto de realismo. Todas estas características demandan una potencia de cálculo extremadamente grande.

Como solución para esta problemática, se plantea la utilización de una arquitectura de computación en paralelo y distribuida, para lograr la mayor potencia de cálculo posible y al mismo tiempo, económicamente viable. Dicha solución consiste en un sistema de distribución de cómputo específicamente diseñado para problemas de renderizado de imágenes tridimensionales orientado a la representación gráfica de volúmenes en tiempo real, a través del uso de múltiples computadoras con placas gráficas programables, interconectadas a través de una red Ethernet de alta performance.

3. Introducción

Visualizar en tiempo real grandes volúmenes de información requiere de la inclusión de nuevas técnicas en el procesamiento de la información, todas ellas tendientes a lograr una mejora, no sólo respecto a las características de la visualización, sino también en el tiempo involucrado en el proceso. Ante estas necesidades y teniendo en cuenta las posibilidades de optimización derivadas de la aplicación de programación paralela y distribuida en la solución del problema, como así también de las brindadas por las nuevas tecnologías en GPU y redes de computadoras. Los tópicos involucrados en esta línea de investigación son:

- *Técnicas de visualización científica*

  La representación tridimensional generada por los escáneres de MRI o CT consiste en múltiples imágenes bidimensionales correspondientes a cortes (generalmente en el plano transversal) de la anatomía del paciente. Estas imágenes se combinan para crear un volumen tridimensional de unidades de información denominados voxel (volumetric pixels), cada una de las cuales, por lo general, tiene una magnitud (nivel de gris) proporcional a la densidad del tejido escaneado. Por ejemplo en CTs, tejidos blandos (como

la piel) se representan con niveles de grises más bajos (oscuros), mientras tejidos rígidos (como el hueso) lo hacen con niveles altos más altos (más claros).

La visualización científica de datos volumétricos se divide en dos grandes tipos de algoritmos de renderización: poligonal y volumétrico.

La renderización de superficie poligonal se basa en el método Marching cubes [6], el cual genera una malla de polígonos extrayendo los vóxeles de densidad similar de un volumen tridimensional. Si bien este método es mucho más rápido para visualizar en tiempo real, tiene la gran desventaja de no permitir la visualización de la información completa del volumen, sino sólo su superficie.

La renderización volumétrica, por su parte, se divide en dos grandes grupos[25]: *Object Order* y *Image Order*. El método *Object Order* toma como punto de partida cada voxel del objeto volumétrico, realizando la proyección sobre el plano 2D de visión. *Image Order* en cambio realiza la proyección utilizando como punto de partida cada píxel del plano 2D de visión. Uno de los algoritmos más utilizados de este último método es denominado *Ray Casting*, el cual fue implementado con excelentes resultados en GPU [26].

- *Técnicas de renderización distribuida*

  Respecto a la renderización paralela distribuida en tiempo real, existe una clasificación, la taxonomía de Molnar [2, 11], la cual básicamente establece dos tipos de técnicas basadas en la manera en cómo distribuye el trabajo entre los diferentes nodos de un sistema distribuido. Estas dos técnicas se conocen con el nombre de *Sort first* o *2D* y *Sort last* o *DB*, sus principales características son:

  - La renderización paralela distribuida *Sort first* o *2D* propone la división de la pantalla en áreas no solapadas, asignando una a cada nodo o unidad de cómputo disponible. De esta forma, cada nodo renderiza una sección determinada de la pantalla completa. Generalmente los algoritmos de renderización utilizados en este caso son del tipo *Image Order*, existen algunos desarrollos en sistemas con memoria compartida [16] y en sistemas GPU-CPU con memoria distribuida [1]. Si bien en ambos casos se logró un buen desempeño, la resolución de las imágenes no fue muy alta. Finalmente con resolución de imágenes muy superiores, así como también volúmenes de gran tamaño, se lograron excelentes resultados [22]. Aunque es un método que requiere poca interacción entre los nodos del sistema, no aprovecha la potencia de cómputo del sistema producto del factor de solapamiento en el cómputo [14, 19].
  - En la técnica *Sort last* o *DB*, la renderización es hecha dividiendo sectores de volúmenes o primitivas, aplicarles la renderización y finalmente realizar la composición de los resultados parciales para obtener la imagen final. Este método es especialmente adecuado para algoritmos de renderizado *Object Order* para resoluciones de la pantalla de salida no demasiado altas y grandes tamaños de volúmenes [7, 4]. Esta es una técnica altamente escalable con una desventaja importante, la composición final implica la interacción entre los diferentes nodos del sistema.

A partir de la taxonomía de Molnar, se realizaron varias propuestas, una de ellas se expone en [20] donde se plantea una técnica híbrida entre *Sort-first* y *Sort-last*, teniendo en cuenta las ventajas de cada uno.

La técnica *Alternate Frame Rendering* (*AFR*), muy utilizada para paralelizar varias GPUs en una única PC [12], es otro tipo de renderización distribuida, la cual se basa en la distribución de frames. Esta técnica muestra una notable escalabilidad, pero agrega latencias entre la entrada y la visualización parcialmente compensada por un alto frame rate. Es por ello que no siempre es un método apto para aplicaciones de tiempo real. Sin embargo hay trabajos de distribución en tiempo real que presentan un buen desempeño comparándolo con los métodos Sort-first/Sort-last [10].

Otros ejemplos de trabajos realizados en esta área son [8, 9, 13, 17, 23, 21] entre otros.

- *Unidad de procesamiento gráfico, GPU*

  Las GPU (Graphics Processor Unit) inicialmente usadas para aplicaciones específicas, como aceleración tridimensional, han evolucionado en los últimos años, permitiendo su uso en un sinnúmero de aplicaciones computacionales altamente paralelizables permitiendo ser una alternativa válida a las supercomputadoras. El éxito de su notable expansión se basa fundamentablemente en su alta potencia de cálculo, su bajo costo, su reducido consumo relativo a su poder computacional y la posibilidad de complementarse con CPU de arquitecturas de amplia difusión tales como ia32 y amd64. Esto permite que la GPU se comporte como un coprocesador para tareas altamente paralelas, mientras que el código menos paralelizable se siga ejecutando sobre CPUs.

  El bajo costo de las GPUs actuales se debe a que utilizan tecnologías con el mismo grado de desarrollo que las utilizadas en las CPUs convencionales [18]. Sin embargo, las GPUs modernas poseen una arquitectura de hardware completamente diferente a las CPUs, permitiendo ejecutar en forma paralela una gran cantidad de cálculos. Están constituidas de un número variable de streaming multiprocessors (SMs) compuesto cada uno de ellos por una cantidad fija de Scalar processor (SPs). Cada SP del SM puede ejecutar simultáneamente la misma operación sobre registros de memoria diferentes de tal forma que un SM se comporta como una arquitectura Single Instruction Multiple Data (SIMD)[3]. Además, cada SM es capaz de realizar una transferencia desde o hacia memoria RAM de la GPU con un ancho de bus mucho mayor a las arquitecturas ia32 convencionales lo que le permite tener una gran potencia de cálculo incluso cuando los datos de entrada superan la capacidad interna de registros de la GPU.

  Los métodos utilizados en la fabricación de GPUs poseen por lo general el mismo nivel de integración que las CPUs actuales con la diferencia fundamental que distribuyen un porcentaje mayor del área del chip para ALUs y controladores de memoria, y menor para caches [5, 15].

  Para poder obtener una escalabilidad apropiada en la solución de los problemas sobre GPU y resolver problemas con complejidades crecientes a un reducido costo, resulta apropiado distribuir la computación GPU sobre varias conectadas a través de una red. Esto representa un desafio mayor, no sólo derivado de la posibilidad de tomar ventajas de varias GPU sino también de los inconvenientes propios de las redes locales, como por ejemplo la latencia y el costo de las comunicaciones respecto a las soluciones integrados en un solo dispositivo físico.

  Comparadas con las supercomputadoras clásicas (mainframes), la utilización de múltiples pequeñas computadoras con CPU+GPU conectadas a redes de alta performance proveen las siguientes ventajas:

  - Alto poder de cálculo comparable con supercomputadoras
  - Bajo costo de adquisición del hardware
  - Bajo costo de mantenimiento (dada la gran compatibilidad del hardware utilizado)
  - Alta escalabilidad
  - Bajo consumo
  - Reducido costo de programación (debido a la reutilización del código no paralelizable y baja obsolescencia del código paralelizable)

  Es interesante destacar que hasta el momento la supercomputadora mas rápida del mundo[24], Tianhe-1A, es un cluster construido a través de PCs con GPUs conectadas por una red de alta performance, lo cual sin duda es un indicador de la potencialidad y competitividad de las GPUs utilizadas en sistemas distribuidos para su uso en la supercomputación moderna.

4. Líneas de Investigación y Desarrollos

Como los clusters GPU-CPU existentes, por lo general, son basados en GPUs de diversas generaciones y fabricantes (heterogéneos) y con redes

Ethernet 100Mbps o 1Gbps, se pretende su utilización para la renderización paralela y distribuida de volúmenes. Para ello se requiere analizar los diferentes algoritmos de renderizado volumétricos utilizados (image-order y object-order) y las diferentes técnicas de procesamiento de la información (Sort-last, Sort-first, Alternating frame) existente, determinando posibles nuevas alternativas o combinaciones de las existentes para obtener una solución óptima del problema dado.

Además, para un máximo aprovechamiento de la arquitectura determinado por un rendimiento de potencia de cómputo muy superior a la CPU, y reducir los efectos no deseados de la misma caraterizados por una latencia de instrucción superior a la CPU, no basta con ejecutar una cantidad de hilos (threads) igual a la cantidad de cores del hardware utilizado, sino que se debe ejecutar miles de hilos simultáneamente [5]. Es por ello que para aprovechar cada una de las características de las GPUs, es importante diseñar algoritmos altamente paralelos, lo cual representa uno de los ejes principales del trabajo propuesto.

5. Resultados obtenidos / esperados

El principal aporte de esta línea de investigación es analizar la factibilidad de utilizar la arquitectura cluster de GPU-CPU para desarrollar soluciones de alto desempeño para visualizacion en sistemas distribuidos de datos volumétricos, estableciendo los límites del modelo en tiempo real. Actualmente se están evaluando distintos parámetros de rendimieto de las GPU, tal como potencia de cálculo y accesos a memoria global. Ésto nos permitirá realizar una estimación del desempeño de los futuros desarrollos.

6. Formación de Recursos Humanos

Los resultados esperados respecto a la formación de recursos humanos son hasta el momento un doctorado en desarrollo; así como también varios trabajos de fin de carrera de la Licenciatura en Ciencias de la Computación.

Actualmente se ha obtenido una beca otorgada por la Facultad de Ciencias Físico, Matemáticas y Naturales de la Universidad Nacional de San Luis, categoría postgrado.

Referencias

[1] Chandrajit Bajaj, I. Ihm, S. Park, and D. Song (2000). Compression-based ray casting of very large volume data in distributed environments. In HPC '00: Proceedings of the The Fourth International Conference on High-Performance Computing in the Asia-Pacific Region-Volume 2, pages 720-725.

[2] Michael Cox. Algorithms for Parallel Rendering. PhD thesis, Department of Computer Science, Princeton University, 1995.

[3] Felipe A. Cruz (2011) Introduction to GPU computing - Nagasaki Advanced Computing Center- Nagasaki University, Japan- PASI-Scientific Computing the Americas-January 2011

[4] Klaus Engel, Markus Hadwiger, Joe M.Kniss, Christof Rezk-Salama, Daniel Weiskopf. - eal-Time Volume Graphics..A K Peters, Ltd, 2006

[5] David B. Kirk, Wen-mei W. Hwu (2010) - Programming Massively Parallel Processors: A Hands-on Approach

[6] William E. Lorensen, Harvey E. Cline, Marching Cubes: A high resolution 3D surface construction algorithm, Computer Graphics, Vol. 21, Nr. 4, July 1987

[7] Stephane Marchesin, C. Mongenet, and J.-M Dischler (2008). Multi-gpu sort-last volume visualization. In Eurographics Symposium on Parallel Graphics and Visualization (EGPGV08).

[8] Stephane Marchesin and Kwan-Liu Ma (2010) Cross-Node Occlusion in Sort-Last Volume Rendering In Proceedings of Eurographics Parallel Graphics and Visualization Symposium (EGPGV)

[9] John McCorquodale and S.V. Lombeyda (2003). The volumepro volume rendering cluster: A vital component of parallel end-to-end solution. Technical report, California Institute of Technology.

[10] Cristinel Mihai Mocan ,Dorian Gorgan , Cluster Based Modeling and Graphical Visualization of Interactive Large Spatial Data. IEEE MIPRO 2010, GVS - Grid and Visualization Systems Conference, May 24th-28th 2010, Opatija, Croatia. Proceedings Vol.I., MEET and GVS, ISBN 978-953-233-051-9, pp. 293-298 (2010)

[11] Steven Molnar, Michael Cox, David Ellsworth, and Henry Fuchs. A Sorting Classification of Parallel Rendering. IEEE Computer Graphics and Algorithms, pages 23-32, July 1994.

[12] Jordi Roca Monfort, Mark Grossman (2009) Scaling of 3D Game Engine Workloads on Modern Multi-GPU Systems - The 1st ACM Conference on High Performance Graphics (HPG-09) August 2009

[13] Kenneth Moreland, Brian Wylie, Constantine Pavlakos (2001). Sort-last parallel rendering for viewing extremely large data sets on tile displays. Paper presented at the Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics.

[14] Carl Mueller. The sort-first rendering architecture for high performance graphics. In ACMSIGGRAPH Computer Graphics (Special Issue on 1995 Symposium on Interactive 3-D Graphics), 1995.

[15] Nvidia, (2008) NVIDIA CUDA Compute Unified Device Architecture - Programming Guide Versi/ón 2.0

[16] Michael E. Palmer, Brian Totty and Stephen Taylor (1998). Ray casting on shared-memory architectures: Memoryhierarchy considerations in volume rendering. IEEE Concurrency, 6(1):20-35

[17] Brian Paul (2004). Chromium for Cluster Rendering Retrieved May 20, 2009, from http://graphics.stanford.edu/ mhouston/VisWorkshop04/ChromiumVis2004pt1.pdf

[18] Cristian Perez, Fabiana Piccoli (2010) - Estimaci/ón de los Par/ámetros de rendimiento de una GPU. Mec/ánica Computacional, Volume XXIX. Number 31. High Performance Computing in Computational Mechanics

[19] Rudrajit Samanta, Thomas Funkhouser, Kai Li, and Jaswinder Pal Singh. Sort-first parallel rendering with a cluster of pcs. In SIGGRAPH 2000 Technical sketches, August 2000.

[20] Rudrajit Samanta et al. Hybrid Sort-First and Sort-Last Parallel Rendering with a Cluster of PCs. SIGGRAPH/Eurographics Workshop on Graphics Hardware. Interlaken, Switzerland, August 2000.

[21] Nicholas Schwarz, S. Venkataraman, L. Renambot , N. Krishnaprasad, V. Vishwanath, J. Leigh, A. Johnson, G. Kent,, and A. Nayak (2004). Vola-tile - a tool for interactive exploration of large volumetric data on scalable tiled displays. In VIS '04: Proceedings of the conference on Visualization '04, page 598.19.

[22] Nicholas Schwarz, Jason Leigh Distributed volume rendering for scalable high-resolution display arrays, GRAPP 2010 - International Conference on Computer Graphics Theory and Applications

[23] Peter Schroeder (1993). Data Parallel Volume Rendering Algorithms for Interactive Visualization. The Visual Computer, 9, 405-416.

[24] Top 500 Supercomputer Sites http://www.top500.org/list/2010/11/100

[25] Daniel Weiskopf, "GPU-Based Interactive Visualization Techniques", 2006, pp.17-33.

[26] Yue Zhao, Xiaoyu Cui,Ying Cheng. High-Performance and Real-Time Volume Rendering in CUDA, Biomedical Engineering and Informatics, 2009. BMEI '09. 2nd International Conference on