Nexo Revista Científica
Vol. 25, No. 02, pp.39-46/Diciembre 2012

ISSN-L 1818-6742

http://www.lamjol.info/index.php/NEXO

Principios y Campos de Aplicación en CUDA
Programación paralela y sus potencialidades

Iván Rivera[1]; M. Vargas-Lombardo[*2].

[1]Licenciatura en Ingeniería en Sistemas y Computación
Facultad de Sistemas Computacionales
Centro Regional Universitario Tecnológico de Azuero
ivan.rivera@utp.ac.pam

[2]Grupo de Investigación de Salud Electrónica y Supercomputación (GISES)
Centro de Investigación, Desarrollo e Innovación en Tecnologías de la Información y las comunicaciones (CIDITIC)
Universidad Tecnológica de Panamá, Panamá
miguel.vargas@utp.ac.pa

(Recibido/received: 14-Junio-2012; aceptado/accepted: 16-Noviembre-2012)

RESUMEN

La siguiente monografía es un compendio de información sobre la tecnología CUDA de procesamiento paralelo en la GPU, para uso computacional en general, descripción de la misma, como también algunas aplicaciones.

Palabras claves: Computación Paralela; CUDA; CPU; GPU.

ABSTRACT

The following monograph is a summary of information regarding GPU parallel processing via CUDA technology for general purpose computations, its description, as well as some applications.

Keywords: Parallel computing, CUDA; CPU; GPU.

[*] Autor para la correspondencia

INTRODUCCIÓN

Las GPUs están cobrando cada vez más importancia por su gran potencia de cálculo, escalabilidad y bajo costo. Para [11],[12],[13] las GPU son unidades de procesamiento gráfico que han evolucionado hasta convertirse en sofisticados procesadores, con un rendimiento orientado a los datos computacionales paralelos, como también a ilimitada horas de cargas de trabajo para la computación científica.

La necesidad de hacer grandes investigaciones que requieren recurso computacional, se ven afectadas por los tiempos de procesamiento, que, se toman en hacer los grandes cálculos y procesamientos científicos. Recientemente, se ha puesto la mirada específicamente en las GPU que promete grandes avances para la computación en general, y a la cual, se le están dedicando muchos esfuerzos para su desarrollo pleno.

Esta tecnología es la computación general sobre unidades de procesamiento gráfico (GPGPU). Actualmente, NVidia es la precursora de esta tecnología y ha creado un nuevo paradigma de desarrollo sobre GPU: CUDA (Compute Unifed Device Architecture) que es un modelo de programación y una arquitectura de cálculo. Esta tecnología aprovecha el procesamiento paralelo y el poder del hardware de la GPU (que solo se pensaba que era para renderización) para realizar operaciones de propósito general, con mayor eficiencia y pasar del tradicional procesamiento central en el CPU a una nueva ventana informática, como es el coprocesamiento. En este documento, se reúne una investigación donde se explica los antecedentes de esta tecnología, qué es la arquitectura CUDA, sus aplicaciones y qué herramienta se ha desarrollado para que la masa de desarrolladores, implemente sobre ella. Además, se describe cómo se realiza el coprocesamiento CPU+GPU y el último producto de esta tecnología: Fermi. Como información ilustrativa, se muestran imágenes sobre este producto, forma en cómo se desarrolla el coprocesamiento y comparativa de rendimiento en este nuevo arquetipo científico.

Este trabajo se organiza de esta forma: en la sección II, se explica sobre la era del procesamiento paralelo. Se continúa con la sección III, en la cual se comentan los aspectos que han dado cabida al auge de la computación sobre GPU. En la sección IV se atiente el paradigma y arquitectura diseñada por NVidia-CUDA. En la sección V, se explica cómo se usa NVidia-CUDA para las aplicaciones científicas. Finalmente, se exponen las conclusiones de este trabajo.

LA ERA DEL PROCESAMIENTO PARALELO

En el procesamiento en paralelo, se puede ejecutar múltiples hilos (threads), en la actualidad del orden de entre 8 y 12. Sin embargo, lo que realmente están haciendo, es fragmentar las aplicaciones secuenciales en pequeños fragmentos de código y asignando su ejecución a los diferentes cores del procesador. Por ejemplo, si en una aplicación que se ejecuta secuencialmente sobre un procesador con múltiples cores, y el desarrollador decidió que algún fragmento se distribuya entre diferentes cores, la aplicación se ejecutará por partes secuenciales en cores diferentes. Hasta que alguna parte termine su trabajo, la siguiente no iniciará. Esto significa que el paralelismo que consigue, en general, un procesador multicore, representa un paralelismo que se ejecuta sobre múltiples aplicaciones en paralelo, pero que normalmente, no ejecuta múltiples secciones de la aplicación en paralelo.

Para conseguir mejorar este objetivo, las GPUs integran lógicas de control más simples y que pretenden ejecutar el máximo de threads en paralelo. Integran memorias cache de menor tamaño, compartidas por diferentes unidades de proceso para evitar tener que acceder a la memoria principal, pero siguen manteniendo una mayor superficie dedicada a las unidades de procesado. Así mismo, integran buses de comunicación de mayor ancho de banda que las CPUs (del orden de 10 veces más rápidos que las CPUs).

Cada vez más desarrolladores de software, tendrán que hacer frente a distintas plataformas de computación paralela y tecnologías que suministren experiencias nuevas y nutridas, dirigidas a crear una base moderna y refinada, destinada a los usuarios.

Unidades centrales de procesamiento

En los últimos 30 años, uno de los procesos significativos para optimizar el rendimiento de dispositivos computacionales de consumo, ha sido el incremento de la velocidad, al cual el reloj del microprocesador opera. Inicialmente, los primeros equipos personales en la década de los años 80, tenían unidades centrales de procesamientos (CPU's) en donde su reloj interno, corría a una velocidad promedio de 1 Megahercio (MHz). Actualmente, 30 años más tarde, gran cantidad de microprocesadores tienen velocidades que están entre 1 a 4 Gigahercio (GHz); 1000 veces más rápidos que los equipos personales de un inicio. Aunque el incremento de la velocidad de los CPU's no es la

única forma de incrementar el rendimiento computacional, es una fuente confiable para mejorar el rendimiento.

Desde otra perspectiva, el mundo del consumo computacional, las supercomputadoras han tenido por décadas formas similares en la obtención de grandes ganancias de rendimiento. El rendimiento de un procesador usado en una supercomputadora ha escalado de manera astronómica, muy parecido a las mejoras en las unidades centrales de computadoras personales. No obstante, además de las mejoras en el rendimiento de un procesador, los manufactureros de supercomputadores han sacado un incremento exponencial en el rendimiento, por el número de procesadores. Ahora bien, no es muy común para los supercomputadores, tener decenas o cientos de miles de núcleos de procesadores, trabajando paralelamente.
En el año 2005, frente a un mercado que se está incrementado competitivamente, pero decrementando en alternativas, los fabricantes de CPU's empezaron a ofrecer procesadores con núcleos, en lugar de uno, como tradicionalmente se hacía. En los años siguientes, se continuó esa tendencia de desarrollo con no solo 3 núcleos, sino, cuatro, seis, hasta ocho núcleos centrales de procesamiento en la unidad. Algunas veces, referido como la revolución de los multinúcleos; esta inclinación ha marcado una transformación en el mercado de la informática de consumo.

PRINCIPIOS DE LA COMPUTACIÓN GPU

Las CPUs, se diseñan empleando sofisticadas instrucciones y lógica de control; y tratan de incrementar el rendimiento de las aplicaciones secuenciales existentes. Para ello, se incorporan memorias cache de mayor tamaño y con menores latencias que eviten el acceso reiterativo a la memoria principal que es mucho más lenta. Además, de esta forma, se consigue evitar el uso reiterado del bus del sistema que es mucho más lento que las comunicaciones de las unidades de aplicación o ejecución del procesador, con las memorias caché integradas también dentro del mismo procesador. Estas mejoras son las que han contribuido al incremento de rendimiento de las CPUs.

Breve historia de los GPU's

Los procesadores han evolucionado, tanto en velocidad de su reloj interno, como en la cantidad de núcleos. Entre tanto, el estado del procesamiento gráfico pasó por un proceso de revolución dramático. A finales de los años 80 y principios de los 90, el aumento de la popularidad de los sistemas operativos gráficos como Microsoft Windows, ayudó a crear un mercado para un nuevo tipo de procesador. Al comienzo de 1990, usuarios empezaban a adquirir tarjetas de aceleramiento gráfico en dos dimensiones (2D), para sus computadoras personales. Esas tarjeras gráficas jugaban un rol como asistentes que ayudaban en operaciones de mapas de bits en sistemas operativos gráficos.

En la misma época, en el mundo de la computación, a un nivel profesional, una compañía de nombre Silicon Graphics, popularizó el uso de gráficas en tres dimensiones (3D) en una variedad de mercados, incluyendo aplicaciones de gobierno, defensa y visualización técnica y científica, así como una forma de proveer herramientas para estudiar efectos cinematográficos.

En 1992, Silicon Graphics, abrió la interfaz de programación para este hardware, realizando la librería OpenGL. Silicon Graphics intentó con esta librería, ser el estándar usado como plataforma independiente, método para escribir aplicaciones gráficas en 3D. Al igual que el procesamiento paralelo y CPU's, solo sería cuestión de tiempo para que todas estas aplicaciones y tecnologías encontraran un camino dirigido hacia las aplicaciones de consumo.

A mediados de 1990, la demanda de aplicaciones empleando gráficos 3D había escalado velozmente, preparando el terreno para dos desarrolladores relevantes. Primero, la realización de juegos en primera persona (first-person), como Doom, Dukem 3D y Quake, ayudó en el inicio de una búsqueda para crear entornos 3D más realistas, para juegos de computadora. Aunque los gráficos 3D trabajarían eventualmente en el camino de juegos de computadoras, la popularidad del género de los juegos de disparos en primera persona, generó la adopción de los gráficos 3D en la informática de consumo. Al mismo tiempo, empresas como NVidia, ATI Technologies y 3dfx Interactive, comenzaron a promocionar aceleradores gráficos, lo suficientemente factible para atraer la atención general. Estos desarrolladores asientan los gráficos en 3D, como una tecnología que figura y promete ocupar un lugar destacado en los próximos años.

Desde el punto de vista del procesamiento paralelo, la liberación de la serie GeForce 3 de NVidia en 2001, representa el más importante avance en la tecnología GPU. La serie GeForce 3 fue la primera en la industria computacional en implementar, el entonces nuevo estándar DirectX 8.0 de Microsoft. Dicho estándar

requería que el hardware compatible contuviera vértices programables arrancables y etapas programables de sombreado de pixeles. Por primera vez, desarrolladores tenían algún control sobre los cálculos exactos que se pueden realizar en las GPU.

Los principios de la Computación GPU

La liberación de los GPU's que tienen múltiples líneas de programación, atrajo a investigadores a la posibilidad de usar el hardware, para algo más que representar OpenGL o DirectX. Durante el enfoque general que se dio en los inicios de la computación GPU fue bastante complicado. Porque los estándares gráficos API, como OpenGl y DirectX, siguen siendo la única forma de interactuar con las GPU; cualquier intento para realizar cálculos sobre una GPU, estaría sujeto a limitaciones de programación dentro de los gráficos API. Por esto, los investigadores exploraron la computación de propósitos generales, a través de gráficos API, tratando de crear que sus problemas aparezcan solucionados en las GPU.

Fundamentalmente, las GPU's de los años 2000 fueron diseñados para producir un color por cada pixel en la pantalla, utilizando unidades aritméticas programables, conocidas como sombreado de pixeles (pixel shaders). En general, un sombreado de pixel usa coordenadas (x,y) para posicionar en la pantalla, como alguna información adicional, y combinar varias entras para determinar el color final. Esta información extra, podría ser: colores de entradas, coordenada de texturas o algún otro atributo que se pasa al sombreado mientras se ejecuta. Pero debido a que la aritmética que se realiza sobre las entradas de colores y texturas, estaba completamente controlada por el programador, investigadores observaron que esas entradas "colores", podrían ser cualquier dato.

Por otra parte, si las entradas eran datos numéricos con algún valor, más que un color, los desarrolladores podrían programar los sombreados de pixeles para realizar los cálculos computacionales sobre estos datos. Los resultados retornaron a la GPU como un pixel final de "color", a pesar de que los colores serían simplemente los resultados de los cálculos, que el programador había dado a las entradas del GPU. Esos datos podrían ser leídos por los investigadores y la GPU nunca volvería a ser la más sabia. En esencia, la GPU estaba siendo engañada en la realización de tareas, haciendo que esas tareas aparecieran como si se tratara de representación estándar. Este engaño o trampa, fue muy ingenioso pero además, muy engorroso.

Debido a la aritmética de rendimiento superior de las GPU's, los resultados iniciales desde esos experimentos, prometían un futuro vislumbrante para la programación en GPU. Sin embargo, el modelo de programación todavía era demasiado restrictivo para la masa de desarrolladores para formarse. Hubo restricciones de recursos, desde que los programas podrían recibir datos de entrada, solo de un conjunto de colores de entrada y texturas. También había limitaciones sobre cómo y dónde, el programador podría escribir resultados en la memoria, ya que los algoritmos requieren de la habilidad para escribir en localidades arbitrarias en memoria y no se podría ejecutar en la GPU. Por otra parte, era casi imposible de predecir, cómo la GPU se ocupara de datos de coma flotante, si el manejo de datos de coma flotante era total, por lo que la mayoría de los cálculos científicos serían incapaces de usar la GPU. Finalmente, cuando el programa calcula los resultados de forma inevitable, falla para terminar o simplemente "cuelga" el sistema; no existía una forma razonablemente buena para depurar el código que estaba ejecutando la GPU.

Como si las limitaciones no fueran lo suficientemente rigurosas, los que pretendían usar una GPU para realizar cálculos generales, tenían que leer y aprender sobre OpenGL o DirectX, ya que eran los medios para lo cual se podría manejar una GPU. No solo significa, que el almacenamiento de datos en texturas gráficas y la ejecución de cálculos esté llamado por funciones de OpenGl o DirectX, sino que escribir los mismos cálculos en lenguajes de programación, solo para gráficos, se conoce como lenguajes de sombreado.

FILOSOFÍA DEL DISEÑO CUDA

No sería hasta cinco años después del lanzamiento de la serie GeForce 3, que la computación GPU estaría lista para el momento estelar. En noviembre del año 2006, NVidia dio a conocer el primer GPU DirectX 10 a la industria: el modelo GeForce 8800 GTX. Este modelo fue además, el primer GPU en ser construido con la arquitectura CUDA de NVidia. Esta arquitectura incluye importantes componentes nuevos, diseñados estrictamente para la computación GPU y su finalidad era aliviar muchas de las limitaciones que impedían las anteriores GPU's, que fueran legítimamente útiles para la computación de propósito general.

La arquitectura CUDA

A diferencia de las previas generaciones que segmentaban los recursos computacionales entre los sombreados (shader), en la computación gráfica, es un

programa de computadora utilizado primordialmente para calcular efectos en el despliegue de imágenes en hardware de gráficos con cierto grado de flexibilidad de vértice y de pixeles. La arquitectura CUDA incluyó la tubería (hilo de ejecución) unificada de sombreadores, permitiendo que todas y cada una de las unidades aritméticas lógicas (ALU) en el chip, a ser comandadas por el programa que intenta realizar cálculos de propósito general. Debido a que NVidia pretende que esta nueva familia de procesadores gráficos sean usados para la computación general, estas ALU's fueron creadas con los estándares de la IEEE para aritmética de punto flotante de precisión simple y diseñadas para usar un conjunto de instrucciones, adaptados para la computación general y no específicamente para gráficos. Además, las unidades de ejecución en la GPU, permiten la lectura aleatoria y el acceso a escritura en la memoria como un buen acceso a un software administrado en caché, conocido como memoria compartida (shared memory). Todas estas características de la arquitectura CUDA, fueron agregadas en medida para crear un GPU, que se destacara en cálculos, además de un rendimiento adecuado en tareas gráficas adicionalmente.

Usando la arquitectura CUDA

A pesar del esfuerzo de NVidia de suministrar a los consumidores un producto para computación y gráficas, no podía dejar de producir hardware, incorporando la arquitectura CUDA. Independientemente de cuántas características hayan añadido NVidia a sus circuitos integrados para facilitar la computación, no seguía existiendo una forma de acceder a estas funcionalidades sin que fuera requisito, OpenGl o DirectX. Esto no solo ha requerido que lo usuarios continúen ocultando sus cálculos como problemas gráficos, sino que necesitaría seguir escribiendo sus cálculos en lenguaje orientado de gráficos como GLSL de OpenGl o HLSL de Microsoft.

Para alcanzar el máximo número de desarrolladores posibles, NVidia tomó el estándar C y agregó un pequeño número de palabras claves (keywords) con el fin de emplear algunas características de la arquitectura CUDA. Algunos meses después de haberse lanzado la GeForce 8800 GTX, NVidia hizo público un compilador para este lenguaje: CUDA C [1]. Con ello, CUDA C se convirtió en el primer lenguaje especialmente diseñado por una compañía de procesadores gráficos, para facilitar la computación general sobre los GPU's.

Además de crear un lenguaje para escribir códigos fuentes para la unidad gráfica, NVidia también provee un controlador de hardware especializado para explotar masivamente la potencia de cálculo de la nueva arquitectura CUDA. Para los usuarios, ya no es necesario tener algún conocimiento sobre interfaces de programación gráficas en OpenGL o DirectX; ni tampoco se requiere forzar el problema para que luzcan como tareas gráficas de computadoras.

APLICACIONES EN CUDA

Desde que debutó a principios de 2007, diversidad de empresas y aplicaciones han disfrutado mucho de un gran éxito por haber escogido CUDA y C como plataforma para crear sus aplicaciones. Estos beneficios a menudo incluyen órdenes de aumento de rendimiento en los últimos estados de la implementación de la técnica. Además, las aplicaciones que se ejecutan sobre procesadores gráficos NVidia, gozan de un rendimiento superior por dólar y por vatios, que las implementaciones construidas exclusivamente para tecnologías de procesadores centrales. Las siguientes aplicaciones [1],[12] son algunas de las maneras en las que los desarrolladores de software han puesto al lenguaje C y a la arquitectura CUDA dentro de un uso exitoso.

Imágenes Médicas

El número de personas que han sido afectados por el cáncer de mama, ha ido en aumento en los últimos 20 años. Gracias en gran parte a los esfuerzos de muchos, la concientización e investigación de la prevención y cura de esta enfermedad, también ha aumentado en los últimos años. Finalmente, todos los casos de cáncer de mama que se detectan a tiempo, son suficientes para prevenir los efectos destructores de la radiación y quimioterapia, las huellas que dejan las operaciones y las consecuencias severas de los casos que no responden a los tratamientos. Como Resultado, investigadores como Sanders [1] comparten un fuerte interés de encontrar rápida, precisa y mínimamente los primeros signos de cáncer de mama.

La mamografía, uno de los mejores métodos para la detección temprana del cáncer de mama, tiene significativas limitaciones. Dos o más imágenes necesitan tomarse y la película debe ser desarrollada y leída por un médico calificado para identificar los potenciales tumores. Adicionalmente, el proceso de rayos X, implica los riesgos de que las repetidas radiaciones queden en el pecho del paciente. Después de un estudio minucioso, los médicos suelen necesitar más información, imágenes más precisas y en ocasiones, hasta una biopsia en un intento de eliminar la

posibilidad del cáncer. Estos errados resultados positivos incurren en costosos trabajos de seguimiento que conllevan a crear un estrés innecesario en el paciente, hasta que se puedan sacar las conclusiones finales.

Las imágenes ultra sonido son más seguras que las imágenes por rayos X, por lo que los médicos optan por usar esto en conjunto con mamografías para ayudarse en el cuidado y diagnóstico del cáncer. Pero los convencionales ultra sonidos, también tienen sus limitaciones. Como resultado de esto, nace Sistemas médicos TechniScan. TechniScan ha desarrollado un prometedor método de ultra sonido tridimensional, mas esta solución no puede ser puesta en práctica, por una simple razón: limitaciones computacionales (cálculo). En pocas palabras, la conversión de los datos del ultra sonido, se concentran en el cálculo de imágenes en tres dimensiones y el mismo requiere tiempo, además que resulta costoso para el uso práctico.

La introducción del primer GPU basado en arquitectura CUDA de NVidia, junto con la programación en C para CUDA, provee una plataforma en la que TechniScan podría transformar los sueños de sus fundadores en realidad. Como su nombre lo indica, su sistema de imagen ultra sonido Svara, usa ondas ultrasónicas para la imagen del pecho del paciente. El sistema Svara de TechniScan, se basa en dos procesadores Tesla C1060 de NVidia, con el fin de procesar 35 gigabytes (GB) de data generados por un reconocimiento de 15 minutos. Gracias a la potencia computacional del modelo Tesla C1060, con solo veinte minutos basta para que el doctor pueda manipular a gran nivel, la imagen tridimensional del cáncer de mama en la mujer.

Dinámica de Fluidos Computacional

Por algunos años, el diseño de rotores eficientemente altos y cuchillas, sigue siendo un arte oscuro. El movimiento asombroso del aire y los fluidos alrededor de estos dispositivos o maquinarias, no pueden ser eficientemente modelados por formulaciones simples o tradicionales; así, las simulaciones precisas resultan demasiado costosas computacionalmente hablando. Solo de los grandes supercomputadores en el mundo, podía esperar recursos de cómputo a la par de modernos sistemas numéricos requeridos para desarrollar y validar los diseños. Desde que son pocos los que tienen acceso a este tipo de máquinas, la innovación en el diseño de estas máquinas seguía paralizado.

La Universidad de Cambridge, en buena tradición iniciada por Charles Babbage, es el hogar para la investigación activa dentro de la computación paralela avanzada. El Dr. Graham Pullan y el estudiante de Doctorado Tobias Brandvik [16] del grupo “many-core”, han identificado correctamente el potencial de la arquitectura CUDA para acelerar la dinámica de fluidos sin precedentes. Sus primeras investigaciones indicaron que los niveles aceptables de rendimiento que podía entregar el potencial de la GPU, se podían realizar en estaciones de trabajo personales. Después, el uso de un pequeño cluster de GPU, superó fácilmente a sus supercomputadores costosos y además, ratificó sus sospechas de que las capacidades de las GPU de NVidia, coincidían con los problemas que pretendían resolver.
Para los investigadores de la Universidad de Cambridge [16], las mejoras masivas que rendimiento que ofrecía C para CUDA representan más que simple aumento progresivo en sus recursos de supercomputación. La disponibilidad del cómputo GPU por su bajo costo, facilita las investigaciones en Cambridge para llevar a cabo, rápida experimentación. Recibiendo resultados experimentales con segundos de diferencia entre cada uno, el proceso de retroalimentación en que los investigadores se basan, ayudan a llegar a grandes avances. Como resultado, el uso de cluster’s de GPU ha transformado fundamentalmente el enfoque de sus investigaciones. Casi la simulación interactiva ha desencadenado nuevas oportunidades para innovar y crear en un nuevo campo de investigación.

Ciencia Ambiental

El incremento de la necesidad de bienes de consumo ecológicos, ha surgido como fruto natural de una industrialización a escala rápida en la economía mundial. La ascendente preocupación por el cambio climático, los precios exorbitantes del petróleo y el creciente aumento de los contaminantes en el aire y agua, ha puesto en un nivel preocupante, los daños colaterales que han tenido esos avances en la producción industrial. Los detergentes y los agentes limpiadores, han sido por largo tiempo, algunos de los productos de consumos más necesarios en los hogares e industrias de uso regular. Como resultado, algunos científicos han empezado a explorar métodos para reducir el impacto ambiental de muchos detergentes sin reducir su eficiencia. Sin embargo, conseguir algo de la nada puede ser una proposición difícil.

Los componentes clave para los agentes de limpieza, son conocidos como surfactantes o tensoactivos. Las

moléculas del surfactante, determinan la capacidad de limpieza y la textura de detergentes y champús, pero a menudo es implicado como el componente más dañino del medio ambiente de los productos de limpieza. Estas moléculas se ligan a la suciedad y después combinadas con agua, los surfactantes pueden ser enjuagados junto con la suciedad. Tradicionalmente, el valor de medición de la limpieza de un nuevo surfactante, requeriría exhaustivas pruebas de laboratorio que involucren numerosas combinaciones de impurezas y materiales para ser limpiados. Este proceso no es de sorprender, puede ser muy pausado y costoso.

La Universidad de Temple ha estado trabajando con la empresa líder Procter & Gamble [15], para usar la simulación interactiva de moléculas de surfactantes o agentes tensoactivos, con suciedad, agua y otros materiales. La introducción de simulación por computadora, no solo ayuda a estimular el enfoque tradicional de laboratorio, sino que aumenta la capacidad y amplitud de las pruebas a numerosas condiciones ambientales variables, más de las que se podría practicar en el pasado. Estos investigadores usaron un GPU acelerado con Highly Optimized Objectoriented Many-particle Dynamics (HOOMD), software de simulación, desarrollado por el Departamento de Energía del Laboratorio Ames.

Al dividir esa simulación por medio de 2 GPU Tesla de NVidia, fueron capaces de adquirir un rendimiento semejante a los 128 núcleos de CPU de la Cray XT3 y de los 1024 CPU's de la máquina BlueGene/L de IBM. Al aumentar el número de GPU's Teslas en esa solución, ya están simulando interacciones de surfactantes a 16 veces el rendimiento de las plataformas anteriores. Desde que CUDA de NVidia ha reducido el tiempo para completar estas simulaciones completamente de varias semanas a un par de horas, los años que vienen deben ofrecer un dramático aumento en productos que tienen mayor eficiencia y reducido impacto al medio ambiente.

CONCLUSIONES

Al realizar esta investigación, se concluye lo siguiente:

Que la tecnología CUDA permite desarrollar avances en muchos campos de la ciencia, llevando consigo, una nueva manera para desarrollar aplicaciones de software más eficientemente, en donde las operaciones de cálculo científico intensivo que anteriormente tomaba meses y años, se desarrollan en mucho menos tiempo.

Que el coprosesamiento implementado en esta tecnología, permite aprovechar todo el hardware disponible, usando los componentes correctos para ejecutar las tareas de cálculo intesivo que se desean llevar a cabo.

La escalabilidad de CUDA proporciona una ventaja en el desarrollo de aplicaciones e impide que el hardware quede "rezagado" al ejecutar los múltiples hilos de ejecución, proporcionando un alto grado de operabilidad, vital para los investigadores y desarrolladores, sin perder calidad (concepto de escalable).

La clave de esta arquitectura, es aprovechar la unificación computacional, la eficiencia multihilo, la potencialidad del hardware, aplicando múltiples hilos de ejecución de manera simultánea, mediante el desarrollo del lenguaje de programación CUDA, que le brinda al programador, un entorno amigable con algunas palabras agregadas al lenguaje C, adaptado para la nueva arquitectura; entre otras.

AGRADECIMIENTO

A la Universidad Tecnológica de Panamá sede de Azuero por su apoyo como también al Centro de Investigación y Desarrollo TIC.

REFERENCIAS

[1] J. Sanders and E. Kandrot, "CUDA by Example, An introduction to general-purpose GPU programming," pp. 2-11, 2010.

[2] J. Nickolls and W. J. Dally, "The GPU computing Era," pp. 2-10, 2010.

[3] E. Lindholm, *et al.*, "NVidia Tesla: A Unified Graphics and Computing Architecture," vol. 28, pp. 39-55, 2008.

[4] J. Nickolls and D. Kirk, "Graphics and Computing GPUs," pp. A2-A77, 2009.

[5] NVidia. (2009). *NVidia CUDA Programming Guide*. Available: http://developer.download.NVidia.com/compute/cuda/2_3/toolkit/docs/NVIDIA_CUDA_Programming_Guide_2.3.pdf

[6] D. Roger, *et al.*, "Efficient Stream reduction on the GPU," 2007.

[7] D. Horn, "Stream Reduction Operations for GPGPU applications," pp. 573-589, 2005.

[8] C. T. Tung and T. T. Seng, "CUDA Programming on New Generation Graphics Cards," p. 3, 2008.

[9] J. Nickolls and D. Kirk, "Appendix A: Graphics and Computing GPUs," pp. 7,24, 2009.

[10] J. F. Croix and S. P. Khatri, "Introduction to GPU programing for EDA," pp. 3-4, 2009.

[11] J.D. Owens, D., Luebke,Govindaraju N, Harris M, Krüger J, Lefohn AE, Purcell TJ. A survey of general-purpose computation on graphics hardware. Comput. Graph. Forum. 2007;26:80–113.

[12] J.D., Owens, M. Houston, D. Luebke, Green S, Stone JE, Phillips JC. GPU computing. Proc. IEEE. 2008;96:879–899.

[13] M. Garland, S. Le Grand, J. Nickolls, Anderson J, Hardwick J, Morton S, Phillips E, Zhang Y, Volkov V. Parallel computing experiences with CUDA. IEEE Micro. 2008;28:13–27.

[14] B. D. Kirk y We-mei W. Hwu. Programming Massively Parallel Processors. A Hands- on Approach. Morgan Kaufmann. ISBN: 978- 0- 12- 381472- 2.

[15] Humber A, And NVIDIA Corporation. " Will Your Next Shampoo be Developed on GPUs?", *Temple University Researchers Help Procter & Gamble Design Better Models For Cleaning Products; Two NVIDIA Tesla GPUs Beat 1024 CPU BlueGene/L Cluster., 2010.*

[16] Brandvik, T. And G. Pullan: SBLOCK: A Framework for Efficient Stencil-Based PDE Solvers on Multi-core Platforms. CIT 2010: 1181-1188

Miguel Vargas-Lombardo obtuvo su título de Ingeniero de Sistemas Computacionales en la Universidad Tecnológica de Panamá en 1998, a partir de ese mismo año fue profesor a tiempo parcial en el área de Arquitectura y Redes de Computadoras. Actualmente es profesor a tiempo completo en la Universidad Tecnológica de Panamá, sede de Coclé, Penonomé, Panamá; además es doctor por la Universidad Politécnica de Madrid, España en el área de Investigación para el Desarrollo de Sistema Software Complejo. Sus principales líneas de investigación son: Cibermedicina, Telemedicina, Grid Computing, y Cloud Computing

Iván Omar Rivera G obtuvo su título de Ingeniero de Sistemas y Computación en la Universidad Tecnología de Panamá, sede de Azuero en 2011. Sus principales líneas de investigación son: Cibermedicina, Telemedicina, Grid Computing y Cloud Computing.