Aplicación de GP-GPU Computing para la optimización de algoritmos científicos mediante el uso de profiling de hardware.

Nicanor Casas, Graciela De Luca, Daniel Giulianelli, Federico Díaz, Waldo Valiente, Sergio Martín

Universidad Nacional de la Matanza

Departamento de Ingeniería e Investigaciones Tecnológicas

Dirección: *Florencio Varela 1703* - Código Postal: 1754

{ncasas, gdeluca, dgiulian, wvaliente, fdiaz, smartin}@ing.unlam.edu.ar

Resumen

En este artículo presentamos el proyecto de investigación sobre optimización de algoritmos científicos programados para para arquitecturas many-core. Estas arquitecturas proveen un gran potencial para los algoritmos que optimizaremos gracias a su alto nivel de paralelismo y simplicidad. En particular en este proyecto, se utilizará programación de procesamiento general para clusters de computadoras con placas gráficas (GPUs).

En una primera etapa, continuamos la investigación realizada sobre algoritmos de simulación tipo N-Body para su optimización sobre arquitecturas GPU, esta vez mediante el uso de profilers de hardware. Buscamos determinar cuáles son los indicadores que indiquen un potencial para realizar optimizaciones mediante herramientas de profiling que provean información obtenida directamente del hardware utilizado.

Luego, en una segunda etapa, ampliaremos dicha investigación para analizar algoritmos de multiplicación de enteros de tamaño arbitrario (Schönhage-Strassen) y de cálculo trayectorias de satélites (SGP4 y SDP4).

Finalmente, utilizando dicha información de profiling tanto de las GPUs como las CPUs, buscamos elaborar una guía de optimizaciones basadas en profiling para algoritmos científicos –especialmente pensada para clusters de GPUs– que pueda ser consultada por científicos y programadores de otras áreas de la ciencia.

En este artículo se presentan las tres etapas que componen este proyecto de investigación.

Palabras clave: GP-GPU, Profiling, N-Body, SGP4, SDP4, Schönhage-Strassen

Contexto

Esta Línea de Investigación es parte del proyecto “Aplicación de GP-GPU Computing para la optimización de algoritmos científicos mediante el uso de profiling de hardware ”, dependiente de la Unidad Académica del Departamento de Ingeniería e Investigaciones Tecnológicas perteneciente al programa de Investigaciones PROINCE de la Universidad Nacional de La Matanza, el cual es continuación de otros proyectos, en especial del proyecto de “Utilización de tecnologías adaptativas para la optimización del uso de recursos y eficiencia energética en clusters de servidores GPU y CPU”.

Introducción

Las actuales unidades de procesamiento de gráficos (GPU) se han convertido en una potente plataforma con un concepto de heterogeneidad en las arquitecturas de computación de múltiples núcleos. Sin embargo, los dominios de aplicación de GPUs se limitan actualmente a sistemas específicos, estructuras gráficas de gran necesidad de cómputo, en gran parte debido a la falta de manejo "de primera clase" de los recursos de la GPU para sistemas multi-tarea de propósito general.

Los avances más recientes asociados a la tecnología de múltiples núcleos han logrado un aumento de un orden de magnitud en el rendimiento del equipo.

Los ejemplos incluyen las unidades de procesamiento gráfico (GPU), dispositivos de cómputo maduros que mejor abrazan un concepto de funciones heterogéneas de computación de múltiples núcleos. De hecho, la información provista por el sitio TOP500, Sitios de Supercomputación, revelan en noviembre de 2011, que tres de los cinco mayores superordenadores emplean grupos de GPUs como los recursos informáticos básicos.

En definitiva se busca un nuevo conjunto de abstracciones del sistema operativo para apoyar las GPU, además otros dispositivos aceleradores, como recursos de primera clase de computación. Estas nuevas abstracciones, llamados colectivamente la API PTask, permiten apoyar un modelo de programación de flujo de datos.

La investigación actual sobre el uso de profiling y técnicas de optimización sobre unidades de procesamiento gráfico (GPU) es todavía incipiente. Se han desarrollado guías sobre “buenas prácticas” para mejorar el rendimiento de un algoritmo [1] [2], y técnicas específicas para mejorar algoritmos de cierto tipo (por ejemplo: [3] para algoritmos de manejo de grandes cantidades de datos en GPU, y para algoritmos tipo N-Body [4] [5]). Sin embargo, muy pocas publicaciones existentes buscan establecer técnicas de optimización que puedan optimizar algoritmos científicos en general basándose exclusivamente en indicadores de hardware.

El uso de indicadores de hardware provee un gran potencial para la optimización de algoritmos en GPU ya que las herramientas de profiling son agnósticas del problema [6]. Esto permite que el desarrollador pueda encontrar los puntos de ejecución que más demoras generan, y otros factores (por ejemplo, la transferencia de memoria intra-chip) que no podrían conocerse de otra manera. Esto permite determinar rápidamente sobre qué aspecto del código centrarse para la optimización del algoritmo. Aunque, algunas publicaciones han buscado establecer cuáles son los indicadores de hardware que más afectan al rendimiento de un algoritmo en GPU utilizando dichas herramientas [7], no buscan desarrollar técnicas de optimización a partir de ellos.

La computación científica es utilizada en todo tipo de campos, incluyendo la ingeniería, la medicina, la física, la astronomía, y la matemática. Estas disciplinas requieren siempre contar con la máxima potencia de procesamiento posible para realizar sus cálculos. Cuanto menos tiempo demore una computadora en ejecutar un algoritmo científico, mejores resultados se podrán obtener de su uso.

En las últimas décadas se han comenzado a desarrollar arquitecturas de procesamiento nuevas que dejan atrás el enfoque clásico de optimización de algoritmos secuenciales. En ellos, todo el esfuerzo de los programadores se enfoca en cómo realizar menos acciones (ejecutar menos instrucciones) dado un algoritmo. Sin embargo, los avances en la tecnología existente han logrado introducir arquitecturas que permiten la ejecución de algoritmos paralelos. Por lo tanto, el esfuerzo de quienes buscan optimizar algoritmos también debe enfocarse en poder utilizar todo el potencial paralelo.

Ha existido una gran cantidad de publicaciones referidas a la utilización de paralelismo convencional (es decir, una o más computadoras con varios núcleos convencionales) que buscan establecer un paradigma de programación que permita determinar qué prácticas arrojarán siempre mejores resultados. Sin embargo, el problema con las arquitecturas paralelas es su imprevisibilidad. No es posible conocer a priori cómo se ejecutaran las tareas, aun si el código es el mismo. Por lo tanto, respecto al paralelismo convencional, sólo se han recopilado guías de buenas prácticas que, adaptadas a cada problema, pueden optimizar la ejecución de un algoritmo.

Los últimos avances en la tecnología han permitido desarrollar arquitecturas many-core (en inglés, significa: muchos núcleos) en las que son reemplazados los núcleos convencionales de un procesador por una gran cantidad de núcleos mucho más simples. Entre estas arquitecturas se encuentra el modelo de GP-GPU (General Purpose GPU programming) que se propone utilizar en esta investigación. Si bien existen algunas guías iniciales sobre el uso de estas arquitecturas para obtener el mejor rendimiento, todavía ninguna de ellas contempla aprovechar la información provista por los contadores de hardware para mejorar algoritmos científicos. Nuestra investigación propone sentar las primeras bases hacia ampliar el conjunto de recomendaciones (guía de buenas prácticas) sobre su uso, para los casos en los que es posible contar con información sobre los contadores de hardware.

El resultado de nuestra investigación tratará de lograr que científicos y programadores de todas las áreas puedan contar con más técnicas de optimización de algoritmos para las arquitecturas GPU, y de esta manera lograr ejecuciones más eficientes cuando éstos tengan acceso a herramientas de profiling y mediciones de indicadores de hardware.

Líneas de Investigación, Desarrollo e Innovación

Actualmente, nos encontramos trabajando sobre los siguientes aspectos: Implementación en el sistema operativo SODIUM de los diferentes modelos e interfaces de gestión de energía existentes.

Aplicación de tecnologías adaptativas sobre aspectos de un sistema operativo, en conjunto con investigadores de la Universidad Nacional de San Pablo.

Diseño y construcción de un sistema operativo como herramienta didáctica.

Resultados y Objetivos

Los resultados obtenidos hasta el momento son relativos debido al poco tiempo que tiene el proyecto en desarrollo.

Como objetivos

Investigar sobre los diferentes indicadores de hardware y herramientas de profiling existentes para unidades de procesamiento gráfico (GPU), y determinar cuales permiten obtener información sobre el potencial de un algoritmo para ser optimizado.
Desarrollar una guía de optimización de algoritmos científicos programados en lenguajes para procesamiento en unidades de procesamiento gráfico (GPU).
Extrapolar los resultados obtenidos para la evaluación de las técnicas de optimización en clusters de 2 o más computadoras con GPU interconectadas por red.
Implementar con éxito los alcances del proyecto en el sistema operativo SODIUM.
Elaborar un diseño de aspectos reconfigurables a fin de permitir un conocimiento profundo por parte de los alumnos.
Establecer los pasos que son necesarios para la utilización de tecnologías adaptables sobre esos aspectos.

Formación de Recursos Humanos

Se está trabajando en conjunto con los alumnos de la Universidad Nacional de La Matanza para lograr que estos, además de los contenidos teóricos de la materia, puedan adquirir el conocimiento del diseño y construcción de funciones de optimización para el uso de GPU.

La presente línea de investigación forma parte del trabajo que el Ingeniero Federico Díaz se encuentra realizando para su tesis de doctorado y el Ingeniero. Waldo Valiente para su tesis de maestría

Además se incorporaron dos alumnos para realizar la iniciación en investigación. Ambos se encuentran cursando el último año de la carrera correspondiente.

Referencias

[1] NVIDIA CUDA™ Programming Guide Versión 5.0, NVIDIA Corporation, 2012.

[2] NVIDIA CUDA™ Best Practices Guide Version 5.0, NVIDIA Corporation. 2012.

[3] J. Siegel, J. Ributzka, Li Xiaoming, “CUDA memory optimizations for large data-structures in the Gravit simulator”. International Conference on Parallel Processing Workshops. September, 2009. Vienna, Austria.

[4] R. G. Belleman, J. Bedorf, S. P. Zwart, “High Performance Direct Gravitational N-body Simulations on Graphics Processing Units”. New Astronomy, 13(2), 2008, 103-112.

[5] F. G. Tinetti, S. M. Martin “Sequential optimization and shared and distributed memory parallelization in clusters: N-Body/Particle Simulation.” Proceedings of Parallel and Distributed Computing and Systems. November, 2012. Las Vegas, United States.

[6] NVIDIA Nsight™ Visual Studio Edition 3.0 User Guide. NVIDIA Corporation. 2013.

[7] G.L.M. Teodoro, R.S. Oliveira, D.O.G. Neto, R.A.C. Ferreira, “Profiling General Purpose GPU Applications”. 21st International Symposium on Computer Architecture and High Performance Computing. October, 2009. Sao Paulo, Brazil.

[8] F. G. Tinetti, S. M. Martin, F. E. Frati, M. Méndez. “Optimization and parallelization experiences using hardware performance counters”. International Supercomputing Conference Mexico. March, 2013. Colima, Mexico.

[9] Sergio Martin, Fernando Tinetti, Nicanor Casas, Graciela De Luca, Daniel Giulianelli. “N-Body Simulation Using GP-GPU: Evaluating Host/Device Memory Transference Overhead”. XIX Congreso Argentino de Ciencia de la Computación. Octubre, 2013. Buenos Aires, Argentina.

[10] Federico Díaz, Fernando Tinetti, Nicanor Casas, Graciela De Luca, Sergio Martin, Daniel Giulianelli. “Análisis de rendimiento del algoritmo SGP4/SDP4 para predicción de posición orbital de satélites artificiales utilizando contadores de hardware”. XIX Congreso Argentino de Ciencia de la Computación. Octubre, 2013. Buenos Aires, Argentina.

[11] Sze, T. W. (2012, June). Schönhage-Strassen algorithm with MapReduce for multiplying terabit integers. In Proceedings of the 2011 International Workshop on Symbolic-Numeric Computation (pp. 54-62). ACM.

[12] Kelso, T. S. "Real-world benchmarking." Satellite Times 3.2 (1996): 80-82