PROCESAMIENTO PARALELO EN MATLAB. SU APLICACIÓN A MODELOS TEÓRICOS

Fernando J. Lage[1], Javier. J. Palmerio[2], Anibal Damian Coppo[2], Zulma Cataldi[1]
tecno_licmae@yahoo.com.ar; jpalmerio@cae.cnea.gov.ar; coppo@cae.cnea.gov.ar, zcataldi@posgrado.frba.utn.edu.ar

1. Universidad Tecnológica Nacional. Facultad Regional Buenos Aires. Depto: de Electrónica
2. Comisión Nacional de Energía Atómica. Centro Atómico Ezeiza

RESUMEN

El presente trabajo se inscribe como una línea nueva de investigación dentro del procesamiento de imágenes y señales. Trata sobre el uso del procesamiento paralelo en el ámbito de los modelos. El mismo se considera el paso previo a su posterior aplicación al procesamiento de imágenes y señales. Esta comunicación muestra la introducción de procesamiento paralelo dentro del software de aplicación. Se han utilizado las librerías de MatLab (denominadas toolbox) como software de aplicación, además se agrego un toolbox de acceso gratuito. En el presente informe se ha incluido los avances a través de un conjunto de ejemplos que han servido como primera aproximación al rendimiento probable de su uso dentro del proyecto general.

Palabras clave: *Procesamiento paralelo, MatLab, imágenes, simulación, GPU*

CONTEXTO

Esta comunicación es parte del PID: *Procesamiento digital de imágenes radiográficas de baja calidad con análisis por transformada de Fourier y wavelets* 2010. Secretaría de Ciencia, Tecnología e Innovación Productiva. Universidad Tecnológica Nacional. Facultad Regional Buenos Aires. Programa: *Electrónica, Informatica y Comunicaciones.* Acreditado en el Programa de Incentivos. Código: 25/C115.

1. INTRODUCCIÓN

El uso de modelos en el cálculo científico es un enfoque que permite explorar temas teóricos o experimentales, buscando predecir los resultados. A este proceso de lo denomina simulación, "La simulación es el proceso de diseñar un modelo de un sistema real y llevar a término experiencias con él, con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias -dentro de los límites impuestos por un cierto criterio o un conjunto de ellos - para el funcionamiento del sistema" (Shannon, 1966). A lo largo de la historia la relación entre simulación computacional y la experimentación ha ido creciendo y afianzándose, comenzando con el método de "Montecarlo" desarrollado por von Neumann y Ulam (von Neumann, 1947), hasta nuestros días donde *"la simulación se ha empleado con éxito en áreas tan disimiles como reservación excesiva de boletos de avión, desarrollo de nuevos productos, flujo de tráfico, mecánica de la atmosfera o procesos atómicos"* (Chapra, 2007).

Con el correr del tiempo, los modelos se fueron aproximando cada vez más a los sistemas reales, haciéndose sus ecuaciones más complejas y con mayor volumen de datos, incrementando la necesidad de más cálculo. Al *"aumentar la cantidad de elementos de procesamiento tiene la ventaja de aproximarse a los problemas en los que la paralelización es inmediata"* (Tinetti, 1998).

1.1. Modelos de computación en paralelo.

La computación en paralelo depende de dos subsistemas íntimamente relacionados entre sí, hardware y software. Dentro del hardware podemos distinguir la unidad de control, los elementos de proceso y la organización de la memoria.

Flynn (1966), propuso una clasificación general del procesamiento paralelo. El primer esquema denominado SISD[1], único flujo de instrucciones y único flujo de datos, representa un microprocesador trabajando un solo conjunto de instrucciones y otro de datos. El segundo modelo es SIMD[2], único flujo de instrucciones y múltiple flujo de datos, representa un conjunto de procesadores donde

[1] SISD: Single Instruction stream, Single Data stream
[2] SIMD: Single Instruction stream, Multiple Data stream

todos ejecutan el mismo programa pero cada uno con un conjunto distinto grupo de datos. El tercer tipo es el MIMD[3], múltiple flujo de instrucciones y múltiple flujo de datos, varios microprocesadores resolviendo problemas diferentes con conjuntos de datos disimiles. Por último el MISD[4], múltiple flujo de instrucciones y único flujo de datos, se tienen varios microprocesadores en cada uno de ellos se ejecuta distinto código con el mismo conjunto de datos.

La organización de la memoria define la manera en la que los elementos de los procesos almacenan los datos. Existen dos formas básicas de organizar la memoria en una computadora de procesamiento paralelo: Toda la memoria se comparte entre todos los elementos de proceso o cada elemento de proceso tiene su propia memoria.

Las computadoras que trabajan en la modalidad de memoria compartida pertenecen a la arquitectura shared-adress-space. En estas computadoras todos los elementos de proceso tienen acceso a toda la memoria, tanto para leer un dato como para escribir un dato. De esta manera la comunicación entre los elementos del proceso se realiza escribiendo y leyendo la memoria compartida.

Las computadoras que trabajan con memoria dedicada para cada elemento de proceso pertenecen a la arquitectura message-passing. Aquí todos los elementos de proceso están conectados por una red de comunicación. A través de esta red los diferentes elementos intercambian datos. En esta arquitectura los datos son privados para cada elemento de proceso. Un elemento de proceso puede enviar un dato a otro elemento en forma de mensaje a través de la red de comunicación. Esta es la forma en la que se intercambia información en esta arquitectura.

Cuando se trabaja con computadoras para procesamiento en paralelo es necesario tener en cuenta que modelos definen su funcionamiento, ya que estos definen como operan las mismas, la relación entre unidad de control y los elemento de proceso, como se organiza la memoria de acuerdo al uso de las arquitecturas shared-adress-space y message-passing.

En la arquitectura shared-adress-space se presenta el inconveniente por el cual un proceso puede leer o escribir un dato. Para arbitrar esto, en la lectura el sistema puede dejar leer un dato a un elemento de proceso (exclusive read) o a varios (concurrent read[5]). En la escritura de un dato también es posible que el sistema deje escribir un dato a un solo elemento de proceso a la vez (exclusive write) o a varios (concurrent write[6]). Al operar en concurrent read todos los elementos de proceso pueden leer un dato dado. Cuando se ejecuta en forma concurrent write, varios elementos de proceso pueden escribir un dato, el protocolo de escritura determina la forma en la que se determina como se almacenará ese dato. Las formas más comunes son: escritura común, escritura arbitraria, escritura por prioridad y escritura en suma. Escritura común significa que el dato solo se guarda si todas las escrituras simultáneas intentan escribir el mismo valor en el dato. En la escritura arbitraria solo un elemento de proceso puede escribir el dato. La escritura por prioridad, se define un orden de prioridad de los elementos de proceso de forma que solo puede escribir el elemento de mayor prioridad. Finalmente la escritura en suma hace que el dato que se escribe sea la suma de todos los que se están tratando de escribir.

Desde otra perspectiva, el software también juega un rol fundamental en el procesamiento en paralelo, existen dos paradigmas acerca de cómo utilizar una computadora de procesamiento paralelo: El paradigma de programación implícita y el de programación explicita. En la programación explicita el programador define como van cooperar los elementos de proceso para resolver un problema en particular. En este paradigma la tarea del compilador es clara, solo debe realizar aquello que el programador declaró en su programa. En la programación implícita el programador escribe código en forma secuencial, sin preocuparse en paralelizar el mismo. Aquí el compilador debe analizar el código, comprenderlo y decidir que porciones del mismo pueden ejecutarse en forma más o menos independiente del resto del código.

Básicamente las operaciones matriciales y vectoriales se pueden descomponer en varias operaciones escalares iguales sobre los diferentes elementos de la matriz o del vector. Además los resultados de las operaciones escalares son, en general, independientes unos de otros. Esto hace que se puedan enviar estas

[3] MIMD: Multiple Instruction stream, Multiple Data stream
[4] MISD: Multiple Instruction stream, Single Data stream
[5] concurrent read: lectura concurrente
[6] concurrent write: escritura concurrente

operaciones escalares idénticas a los diferentes elementos de proceso. En este contexto el cálculo matricial y vectorial es paralelizable. Esto hace que estas operaciones sean las primeras en recibir los beneficios del procesamiento en paralelo permitiendo acceso al usuario en forma implícita.

1.2. Computación en paralelo con MATLAB.

En las últimas versiones de MATLAB se disponen de dos opciones para realizar cálculo en paralelo. Una opción es trabajar con paralelización implícita. Como ya se mencionó en este modelo se deja que el compilador paralelice el código. En MATLAB esto es un poco diferente, no se paraleliza el código en el momento de correrlo, sino que existen funciones y operaciones ya disponibles para aprovechar el cálculo en paralelo. Esta opción es paralelización implícita.

La otra opción de paralelización en MATLAB es utilizar el toolbox de computación distribuida. Este toolbox se diseñó originalmente para aprovechar las redes de computadoras en donde hay una que distribuye trabajo (planificador) a las demás (trabajadores). Los trabajadores reciben un programa del planificador, lo ejecutan y devuelven el resultado al mismo.

El toolbox de computación distribuida fue evolucionando hasta la actualidad, de modo que permite simular trabajadores locales dentro de una computadora con un procesador de varios núcleos. De esta forma cada núcleo trabaja sobre un programa (o un ciclo for) y devuelve un resultado al terminar. Al trabajar con estos toolboxes el modelo de cálculo se asemeja a MIMD.

Existe una alternativa más para utilizar procesamiento paralelo en MATLAB y es operar sobre la placa de vídeo. NVIDIA es una compañía que hasta hace algunos años se dedicaba principalmente al mercado de palcas de video para videojuegos y para computadoras, actualmente diseña y desarrolla productos basados en una arquitectura multinucleos denominada CUDA, esta arquitectura permite realizar cálculo paralelo aprovechando la gran potencia de la GPU (unidad de procesamiento gráfico), la misma se halla en GPUs de las series GeForce[7] y Tesla entre otras.

La arquitectura CUDA permite disponer de varios elementos de proceso que realizan la misma operación sobre diferentes datos (SIMD). El leguaje de programación para esta arquitectura es similar a C pero posee además un conjunto de funciones propias. Existe para MATLAB un toolbox gratuito (GPUMat) que permite utilizar CUDA y provee versiones paralelizadas de funciones y de operaciones de MATLAB para ser ejecutadas sobre la GPU.

En el presente trabajo se muestran resultados obtenidos en un desarrollo teórico, usando las toolbox de MATLAB con paralelización implícita sobre una CPU Core2Duo, y con GPUMat sobre una placa del tipo GeForce.

2. LÍNEAS DE INVESTIGACION

Dentro del proyecto de procesamiento de imágenes se está trabajando en tres líneas de investigación básicas: a) mejoras de las imágenes por ecualización y filtrado (FFT, y Wavelet), b) optimización de tiempos por procesamiento paralelo y c) búsqueda de anomalías y determinación de patrones por métodos automáticos.

3. RESULTADOS OBTENIDOS/ESPERADOS

3.1. Pruebas de paralelización implícita aplicadas a la solución de conjuntos de Julia (CPU /GPU).

El desarrollo de esta experiencia se basó en un conjunto pruebas sobre los denominados conjuntos de Julia (1914), llamados así por el matemático Gastón Julia (1893-1978) quien fue maestro de Benoit Mandelbrot (1967)(autor del fractal que lleva su nombre). Básicamente son estructuras auto semejantes, fractales. Los fractales están asociados a sistemas dinámicos, y se obtienen en forma iterativa. Estos conjuntos se forman con elementos del plano complejo. Para este trabajo se eligieron siete conjuntos de Julia de la familia $K = K^2 + C$. Donde K es un número complejo que se va recalculando en cada iteración y C es otro número complejo que se mantiene constante durante todo el proceso (o sea se estudian las orbitas de los puntos del plano complejo). Al finalizar las iteraciones, los elementos que

[7] GeForce y Tesla son marcas registradas de NVIDEA

pertenecen al conjunto de Julia son aquellos que no tuvieron una clara tendencia al infinito.

Se utilizaron siete fractales para evaluar la performance del CPU y la GPU en MATLAB a través del toolbox GPUMAT para acceder al cálculo en paralelo sobre la placa de video. Se escribieron dos códigos muy similares, uno que trabaja sobre el CPU y otro sobre la GPU. Las corridas sobre la GPU se realizaron explorando una porción del plano complejo con una cantidad de puntos de 2500^2. Las corridas sobre el CPU se hicieron también sobre la misma porción del plano complejo, pero con una cantidad de puntos de 1250^2.

En la tabla 1 se muestra los resultados obtenidos con los siete fractales y la cantidad de iteraciones utilizadas para cada uno.

C	Tiempo GPU (s)	Tiempo CPU (s)	Iteraciones	%CPU/GPU
-1.3+0.1i	0.7981	6.3653	30	797.5
-0.745429+0i	0.9298	11.2664	20	1211.7
0.27334-0.00742i	1.2519	18.5366	30	1480.7
0.11031031-0.67037i	1.0766	30.1663	30	2802.0
-0.488679-0.56790i	1.0890	28.6787	30	2633.5
-0.561321+0.641000i	1.0747	30.9587	30	2882.5
-0.745429+0.11308i	4.4318	187.6091	128	4233.2

Tabla 1. Resultados de las corridas de los programas sobre los fractales de Julia. Cada fractal está asociado a un numero complejo C.

A modo de ejemplo se incluyen dos gráficos obtenidos con ambos programas (ver gráficos 1 y 2). Las imágenes obtenidas con la GPU poseen el doble de resolución que las obtenidas con el CPU, pero esto no se aprecia debido al tamaño de los gráficos.

Todas las corridas se realizaron en una computadora con Windows XP SP3 y con un procesador Core2Duo con 2GB de memoria RAM DDR2 y una placa de video NVIDIA 9500GT de 512MB de VRAM DDR2.

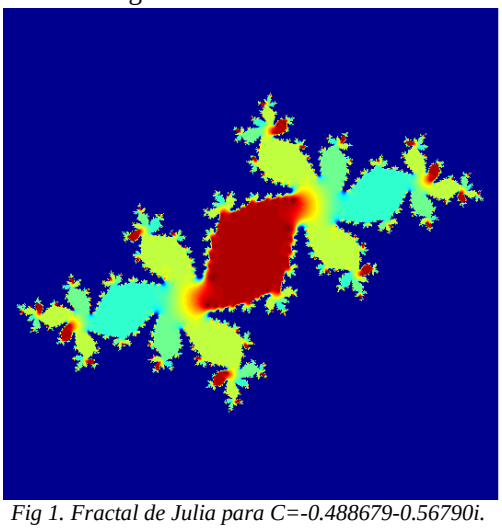

Fig 1. Fractal de Julia para C=-0.488679-0.56790i.

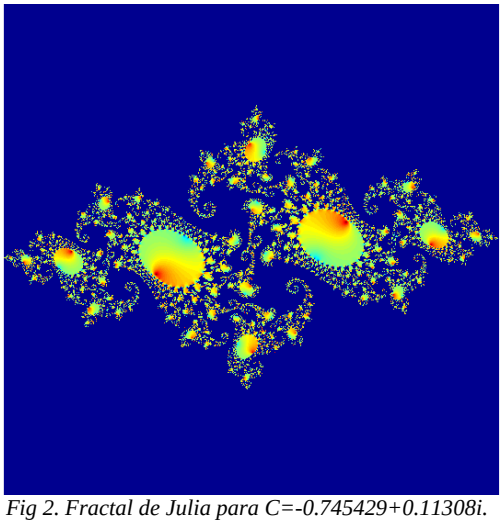

Fig 2. Fractal de Julia para C=-0.745429+0.11308i.

Para mejorar la sensibilidad de las amplitudes de los números complejos cercanas a cero, se ecualizó por medio de la siguiente expresión (ecuación 1):

$$Amp\_Ajustada = e^{-|Z|}$$

Ec 1. Amplitud ajustada. Se utilizó una exponencial decreciente sobre la amplitud original.

Este ajuste permite percibir mejor las pequeñas diferencias de amplitud de los puntos del plano complejo con amplitudes bajas.

3.2. Pruebas con transporte de fotones gamma (CPU cálculo en serie/CPU paralelización explicita)

En este momento se encuentra en desarrollo un conjunto de modelos basados en los métodos de

Monte Carlo que dan la posibilidad de obtener soluciones aproximadas a una gran variedad de problemas mediante el uso de experimentos de muestreo estadístico en una computadora.
El transporte de fotones gamma está controlado por la interacción de los mismos con la materia. La aplicación que se construye simula el transporte de fotones gamma provenientes de una fuente puntual que atraviesan una plancha de Plomo de un centímetro de espesor y luego son colectados en un cilindro de Germanio.
El código se desarrolló para el uso explícito de un ciclo *parfor*, de manera que en una configuración de varios núcleos se pueda resolver un ciclo en cada uno.
El procesamiento en paralelo actualmente está disponible para cualquier usuario de computadoras domesticas. En este trabajo se analizaron varias opciones para trabajar sobre MATLAB. Estas opciones permiten utilizar los modelos de cálculo en paralelo más comunes, SIMD y MIMD. Ninguno de estos modelos es mejor que el otro, simplemente se adecuan a problemas diferentes. Cada una de las aplicaciones elegida explota mejor alguna de las arquitecturas. Los conjuntos de Julia son más adecuados para resolver sobre una arquitectura SIMD mientras que el transporte de fotones es mejor manejada por una arquitectura MIMD.
En ambas aplicaciones se notó un gran incremento de desempeño al utilizar el cálculo en paralelo en lugar del tradicional cálculo en serie.

4. FORMACION DE RECURSOS HUMANOS

La producción del grupo de investigación se plasma en comunicaciones a eventos y artículos (Lage *et al.* 2008, 2009), formación de los investigadores y tecnólogos participantes en el proyecto (Javier Palmerio) y dos tesis de grado en Ciencias Aplicadas (Gabriel Loperena, Daniel Inzillo).

5. BIBLIOGRAFÍA

Lage, F. y Cataldi, Z. 2008. *Procesamiento digital de imágenes radiográficas de baja calidad con onditas: Caso de diagnóstico en pequeños mamíferos.* WICC 2008. La Pampa. 5 y 6 de mayo.

Lage, F. y Cataldi, Z. 2009. *Procesamiento digital de imágenes radiográficas de baja calidad con wavelets para diagnóstico en pequeños mamíferos.* WMSCI 2010 14th World Multi-Conference on Systemics, Cybernetics and Informatics.

Chapra, S. C., Canale, R. P., 2007. Métodos Numéricos para Ingenieros. McGraw Hill, Interamericana, Mexico.

Flynn, M. J., 1966. Very High-Speed Computing Systems. Proceedings of the IEEE, vol. 54, diciembre.

Julia, G. 1917. Mémoire sur l'itération des fonctions rationnelles, Journal de Mathématiques Pures et Appliquées. Paris.

Mandelbrot, B. 1966, How Long Is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension *Science, New Series*, Vol. 156, No. 3775. (May 5), pp. 636-638.

Shannon, R., Johannes, J. D. (1976) "Systems simulation: the art and science" IEEE Transactions on Systems, Man and Cybernetics. Vol. 6(10). pp. 723-724.

von Neumann, J., Ulam, S., Richtmyer, D. (1947). Statistical methods in neutron diffusion, Los Alamos Scientific Laboratory report LAMS-551.

Tinnetti, F. G., De Giusti, A. (1998) Procesamiento Paralelo. Conceptos de Arquitectura y Algoritmos. Ed. Exacta. La Plata. Bs. As. Argentina