PEDECIBA Informática
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

Reporte Técnico RT 09-05

Trabajos preliminares sobre radiosidad y paralelismo

Pablo Ezzatti **Eduardo Fernández**

2009

Trabajos preliminares sobre radiosidad y paralelismo
Ezzatti, Pablo; Fernández, Eduardo
ISSN 0797-6410
Reporte Técnico RT 09-05
PEDECIBA
Instituto de Computación – Facultad de Ingeniería
Universidad de la República

Montevideo, Uruguay, marzo de 2009

Trabajos preliminares sobre radiosidad y paralelismo

Pablo Ezzatti
pezzatti@fing.edu.uy

Eduardo Fernández
eduardof@fing.edu.uy

Instituto de Computación
Facultad de Ingeniería, Universidad de la República
J. Herrera y Reissig 565, Montevideo, Uruguay

Marzo 2009

Palabras claves: iluminación global, eliminación gaussiana, OpenMP.

Resumen

Uno de los métodos para resolver el problema de iluminación global es la técnica de radiosidad, cuyo cálculo es independiente de la ubicación del observador pero que, como contrapartida, posee un alto costo computacional.

La búsqueda por disminuir los tiempos de ejecución del método de radiosidad, para así cubrir problemas más complejos y con mayor precisión, motiva la aplicación de estrategias de computación de alto desempeño. En este reporte se presentan algunos trabajos peliminares en dicha línea, aplicando técnicas de paralelismo de memoria compartida y utilizando OpenMP en la etapa de resolución de los sistemas lineales generados por el método de radiosidad.

1. Introducción

Una de las líneas de trabajo en computación gráfica es el estudio de algoritmos para el cálculo de iluminación, en particular de la iluminación global. Existen diversas técnicas para alcanzar dicho objetivo, una de las más difundidas es la de radiosidad.

Las técnicas de radiosidad tratan de resolver el problema básico de la generación de imágenes realistas. Se basan en el supuesto de que tanto la luz emitida como la reflejada sigue el modelo de iluminación de Lambert [10], siendo por tanto superficies de emisión y reflexión difusas. La resolución del problema de radiosidad por métodos tradicionales implica un alto costo computacional, el cual es mayormente empleado en la resolución de un sistema lineal no disperso de tamaño la cantidad de elementos (denominados parches) en la que se discretizó la escena.

Lo expresado anteriormente motiva el estudio de estrategias de alto desempeño, en particular las técnicas de paralelismo a ser empleadas en los algoritmos de radiosidad.

La aplicación de estrategias de programación paralela se encuentra sumamente ligada al hardware a utilizar. El desarrollo de la computación de alto desempeño (HPC por sus siglas en inglés) ha sufrido varios cambios de rumbo. En los años 80 explotó la investigación en HPC utilizando grandes supercomputadoras de memoria compartida. Luego, en los años 90 surgieron los cluster de memoria distribuida como una alternativa más económica. Esto llevó a una contracción importante de los equipos de memoria compartida. Con posterioridad, en los últimos años, debido a la aparición de equipos dual-core, core 2 duo y quad-core, entre otros, empezaron a extenderse las estrategias híbridas. Estas consisten en la ejecución de aplicaciones en paralelo con estrategias de memoria distribuida en distintos nodos y aplicaciones en paralelo con estrategias de memoria compartida dentro de cada nodo. Este hecho es constatable al observar la evolución de la lista TOP 500 [5].

La nueva explosión en la utilización del hardware de memoria compartida impacta directamente en las herramientas utilizadas. Como ejemplo basta mencionar que la utilización de OpenMP [4] ha tomando un nuevo impulso.

Aunado a lo expresado anteriormente, en el contexto de la computación gráfica es de interés el trabajo relacionado con las tarjetas gráficas (GPU) que con la evolución que han alcanzado en los últimos años se han transformado en verdaderos multiprocesadores de memoria compartida.

Este trabajo busca una mayor comprensión por parte de los autores, de los algoritmos de radiosidad y en particular el estudio de la aplicación de estrategias de programación paralela sobre memoria compartida utilizando OpenMP.

El resto del documento está estructurado de la forma que se describe a continuación.

En la sección 2 se brinda una aproximación al problema de radiosidad, el planteo matemático y los algoritmos de resolución. En la sección 3 se presentan los estudios de tiempos de la versión serial de la eliminación gaussiana, se describe la propuesta y se muestran los tiempos de la versión que incorpora las estrategias de paralelismo. Finalmente, en la sección 4 se describen las principales conclusiones del trabajo realizado y el trabajo futuro a partir de la situación actual.

2. Planteo del problema

En el área de computación gráfica se tratan gran cantidad de problemas referentes a la generación de imágenes por computadoras. Uno de los problemas importantes en el área es el cálculo de la iluminación global de los escenarios o escenas modeladas. En la iluminación global se considera no solamente la interacción entre la fuente luminosa y los objetos sino también las luces indirectas surgidas de la reflexión de estas en los objetos.

Una de las técnicas empleadas para resolver el problema de iluminación global es la técnica de radiosidad, que se utiliza en modelos donde solamente haya superficies de reflexión difusa. Dos de las ventajas de esta técnica es que los resultados tienden a ser físicamente realistas y que los resultados son independientes de la ubicación del observador. Como contrapartida, posee un alto costo computacional. Además, el método es muy sensible al movimiento de los objetos que componen la escena.

Formalmente, la radiosidad de una superficie se define como energía luminosa/unidad de tiempo/unidad de área (W/m^2) y se compone por la radiosidad emitida por la superficie y la radiosidad reflejada proveniente de otras superficies (radiosidad = radiosidad emitida + radiosidad reflejada). En forma matemática esta relación está descrita en la ecuación 1.

$$b_j = e_j + p_j * H_j \tag{1}$$

donde:
B_j radiosidad de la superficie j
E_j radiosidad emitida por la superficie j
p_j coeficiente de reflectividad de la superficie j
H_j radiosidad incidente en la superficie j

Si se subdivide la escena en una cantidad finita de elementos de superficie (o parches) es posible calcular la ecuación de radiosidad para un parche. Es más, la luz que refleja un parche es igual a la luz que recibe multiplicada por el coeficiente de absorción. La luz recibida se expresa en la ecuación 2 [8].

$$H(x) = \int_s B(x')F(x,x')dA' \tag{2}$$

donde:
$B(x')$ = radiosidad en el punto x'.

$$F(x,x') = \begin{cases} 0 & \text{si } x \text{ no ve a } x' \\ \frac{\cos\theta_i \cos\theta_o}{|x-x'|^2} & \end{cases}$$

A' área
$H(x)$ = luz recibida en x

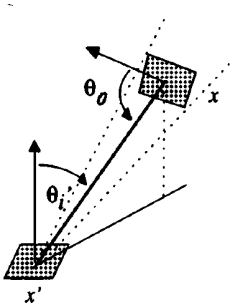

Figura 1: Variables para el cálculo de F

La ecuación 2 en general no es resoluble simbólicamente, pudiéndose resoler en forma numérica a partir de la discretización de los elementos de la escena. Por esta razón, dado que el dominio (escena) modelado ya fue subdividido en parches, es posible transformar la ecuación en algo computable, como se presenta en la ecuación 3, donde la integral ha sido sustituida por una sumatoria y las funciones de la ecuación 2 se discretizan.

$$H_j = \sum_{i=1}^{N} b_i * F_{ij} \tag{3}$$

El término F_{ij} se denomina factor de forma, es una magnitud adimensional que representa la proporción de la radiosidad emitida por el parche i que alcanza al parche j. El cálculo de los factores de forma se puede realizar a través de diversas técnicas, como la utilización de hemicubos [9]. Su cálculo no será desarrollado en este trabajo. Para mayor información el libro *Radiosity and Realistic Image Synthesis* [8] es una excelente referencia.

Si se sustituye la expresión de H_j de la ecuación 3 en la ecuación 1 se puede obtener la ecuación 4.

$$b_j = e_j + p_j \sum_{i=1}^{N} b_i * F_{ij} \tag{4}$$

El resultado que se presenta en la ecuación 4 comúnmente se conoce por ecuación de radiosidad [11]. La resolución del problema de radiosidad para todos los parches implica resolver un sistema lineal $(I - PF)b = e$ donde P es una matriz diagonal que contiene los p_j y F es una matriz que contiene los factores de forma. Esto se puede deducir si en (4) se pasa la sumatoria hacia la izquierda de la igualdad, y la ecuación resultante se desarrolla N veces (una por parche). Los elementos del sistema lineal anterior se pueden observar en la Figura 2.

$$\begin{pmatrix} 1 - p_1F_{11} & -p_1F_{12} & \dots & -p_1F_{1N} \\ -p_2F_{22} & 1 - p_2F_{22} & \dots & -p_2F_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ -p_NF_{N1} & -p_NF_{N2} & \dots & 1 - p_NF_{NN} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix}$$

Figura 2: Desarrollo del sistema lineal empleado para la resolución del problema de radiosidad.

Con este desarrollo, el problema de radiosidad se redujo a la resolución de un sistema lineal de ecuaciones. Resolver de forma eficiente sistemas lineales es uno de los problemas principales en computación científica. La afirmación anterior se sustenta en que es una de las etapas más costosas en tiempo de cómputo en las aplicaciones utilizadas para la resolución de varios problemas complejos de distintas áreas de interés. Uno de los métodos más conocidos para

la resolución de sistemas lineales es la eliminación gaussiana. Su alto costo motiva la aplicación de estrategias de computación de alto desempeño.

En los últimos años la evolución de las tarjetas gráficas o GPUs (Graphics Processing Unit) ha sido importante, posibilitándose de forma progresiva niveles superiores de programación. A la fecha las GPUs no son simples periféricos gráficos sino que se han transformado en verdaderos multiprocesadores de memoria compartida. El relevamiento de Luebke et al. [12] y el sitio web de GPGPU (General Purpose GPU) [3] permiten observar aplicaciones de carácter general demandantes de grandes volúmenes de cómputo, que utilizan las GPU como un coprocesador de cálculo paralelo. Por esta razón, es de interés evaluar las posibilidades de aplicar estrategias de memoria compartida a las GPU, en particular en problemas referentes en el área de estudio.

Sumado a los avances en las GPUs, también han habido avances en las CPU, debido a la aparición de arquitecturas económicas de memoria compartida, en especial con el advenimiento de los procesadores dual-core, quad-core, etc [1, 2].

Si bien en el problema de radiosidad los sistemas lineales contienen matrices esencialmente densas, en otras áreas de conocimiento se generan sistemas lineales dispersos. Como ejemplos se pueden mencionar los métodos de elementos finitos, diferencias finitas y volúmenes finitos. Existen métodos específicos para este tipo de sistemas, en particular en lo referente a la aplicación de las técnicas de paralelismo.

3. Propuesta

En base a lo visto en la sección anterior el trabajo se centró en paralelizar la etapa de resolución de sistemas lineales, utilizando el método de eliminación gaussiana con pivoteo parcial. Para ello se utilizan estrategias de memoria compartida, que son implementadas sobre OpenMP.

La metodología de desarrollo consistió en una primera etapa de estudio de tiempos de ejecución con el fin de detectar zonas de código críticas en cuanto a costo computacional. Los resultados de este estudio permiten atacar las etapas más importantes ya que estas determinan, según la ley de Amdhal [7], el tiempo total del código paralelo. En un segunda etapa, se incorporaron las instrucciones de OpenMP al código y se realizaron diversos experimentos (con distintos tamaños de matrices) en computadores locales. Por último se evaluó la versión en el Superdordenador Virtual Galego (SVG) del Centro de Supercomputación de Galicia (CESGA) [6]. También se evaluaron otras hipótesis de trabajo para la incorporación de estrategias de paralelismo con OpenMP.

3.1. Estudio de tiempos de la versión serial

Para el estudio de los tiempos de ejecución de la versión serial se utilizó un computador Intel dual-core, donde cada procesador es Pentium 4, ejecutando cada uno a 2.8 GHz con 1Gb de memoria RAM, corriendo bajo el sistema operativo Linux distribución Debian. En todos los casos se evaluaron los promedios de cinco corridas independientes. También se presentan las desviaciones estándar de los promedios calculados.

En la Tabla 1 se presentan los tiempos de ejecución, medidos en segundos, de las distintas etapas delimitadas de la escalerización gaussiana para matrices de 512×512.

Etapa	**Tiempo de ejecución** (segundos)	**Desv. Est.**
Inicialización	0,000	0,000
Escalerización	5,322	0,148
Resolución	0,002	0,004

Tabla 1: Tiempos de ejecución de la versión serial para matrices de rango 512.

En base a los tiempos de ejecución presentados en la Tabla 1 se puede deducir que la etapa crucial a la cual aplicar estrategias de alto desempeño es la escalerización, ya que la misma representa más de un 99 % del tiempo total de cómputo de la rutina.

3.2. Implementación

A continuación se menciona la estrategia para la inclusión de instrucciones OpenMP a la escalerización gaussiana. La misma se centra en paralelizar el loop intermedio de la etapa de factorización, línea 15 del Algoritmo 1.

Es necesario destacar que la separación de tareas del loop intermedio permite en forma inmediata un balance de carga. Esto simplifica la tarea, ya que no implica ningún tipo de scheduling particular.

En base a lo mencionado anteriormente, la paralelización consistió en incluir una instrucción *OMP PARALLEL DO* al loop teniendo que duplicar la variable (declararla como privada a cada proceso) donde se almacena la división del elemento de la fila a eliminar sobre el pivote (línea 16 en el Algoritmo 1).

3.2.1. Otras líneas

Además de lo expuesto se estudiaron y evaluaron en forma preliminar otras alternativas buscando mejorar los resultados obtenidos.

Algunas de las ideas estudiadas que no aportaron resultados significativos fueron:

- Paralelizar la inicialización del vector de permutaciones mediante una instrucción *DO PARALLEL*.
- Paralelizar el loop interior de la factorización.
- Paralelizar la agregación en la sustitución.

Otra opción no evaluada sería trabajar por bloques. Esto permitiría desfasar varios cálculos tanto en la etapa de factorización como en la de resolución. Por ejemplo, en la etapa de resolución se podría resolver un bloque de datos e ir reflejando las variables resueltas (columnas) en las distintas filas (bloques de filas) de la matriz.

Algoritmo 1 Pseudo-código de eliminación gaussiana

```
1: for i = 1, n do
2:     piv(i) = i
3: end for
4: for i = 1, n-1 do
5:     magnitud = 0.
6:     for j = 1, n do
7:         if (abs(A(piv(j),i)) >magnitud) then
8:             magnitud = abs(A(piv(j),i))
9:             fila = j
10:        end if
11:    end for
12:    tmp = piv(i)
13:    piv(i)=piv(fila)
14:    piv(fila)=tmp
15:    for j = i+1, n do
16:        t = a(piv(j),i) / a(piv(i),i)
17:        for k = i, n+1 do
18:            a(piv(j),k) = a(piv(j),k) - a(piv(i),k) * t
19:        end for
20:    end for
21: end for
22: for i = n,1, -1 do
23:    for j = i+1, n do
24:        a(piv(i),n+1) = a(piv(i),n+1) - x(j) * a(piv(i),j)
25:    end for
26:    x(i) = a(piv(i),n+1) / a(piv(i),i)
27: end for
```

3.2.2. Tiempos de ejecución de la versión paralela

Al igual que en la sección anterior los resultados que se presentan son el promedio de 5 ejecuciones independientes acompañadas de sus correspondientes desviaciones estándares.

El equipo que se utilizó es un blade DELL PE 1955 con doble procesador quad-core Intel Xeon 5355 a 2.66GHz con 8GB de memoria principal, utilizado en forma dedicada.

En las Tablas 2, 3 y 4 se presentan los tiempos de ejecución de la eliminación gaussiana para matrices de rango 512, 1024 y 2048 respectivamente en el supercomputador SVG del CESGA.

Para evaluar el desempeño computacional de los algoritmos paralelos empleados en este trabajo se utilizan las métricas speedup y eficiencia. El speedup absoluto relaciona el tiempo de ejecución del mejor (mejor en el sentido de tiempo de ejecución) algoritmo secuencial conocido con el tiempo de ejecución del algoritmo ejecutando en paralelo. La definición del speedup absoluto es poco práctica, porque no siempre se conoce el mejor algoritmo serial para resolver un problema, y en caso que éste se conozca, en la mayoría de las ocasiones no se dispone del algoritmo para evaluar su ejecución en un equipo determinado. Por este motivo, en general suele utilizarse como base de la com-

Threads	**Tiempo de ejecución** (segundos)	**Desv. est.**
1	1.169	0,002
2	1,362	0,007
4	0,946	0,002
8	0,695	0,010

Tabla 2: Tiempos de ejecución para una matriz de 512×512.

Threads	**Tiempo de ejecución** (segundos)	**Desv. est.**
1	10,134	0,034
2	5,143	0,016
4	2,646	0,006
8	1,651	0,107

Tabla 3: Tiempos de ejecución para una matriz de 1024×1024.

paración el tiempo de ejecución del algoritmo paralelo trabajando en un único procesador. La medida resultante, denominada speedup algorítmico, se define mediante la expresión de la Ecuación 5, donde $T(1)$ es el tiempo de ejecución del algoritmo paralelo ejecutando en un único procesador.

$$S(n) = \frac{T(1)}{T(n)} \tag{5}$$

En ocasiones no solo es importante el valor de speedup para una cantidad determinada de procesos, sino que también es relevante observar la variación de la ganancia a medida que crece la cantidad de procesadores utilizados. En este caso se puede utilizar para medir la métrica eficiencia, que corresponde al valor normalizado del speedup, relativo al número de procesadores utilizados. La métrica se define por la Ecuación 6, y dependiendo del modo de calcular el speedup se tendrá un valor de eficiencia absoluta o de eficiencia algorítmica (también denominada eficiencia relativa).

$$E(n) = \frac{S(n)}{n} \tag{6}$$

La situación ideal al utilizar un algoritmo paralelo es la de lograr el speedup lineal, es decir al utilizar n procesadores obtener una mejora de factor n en el tiempo de ejecución. Sin embargo, la situación habitual es que utilizando un algoritmo paralelo sobre n procesadores se obtiene una mejora de factor menor que n, situación conocida como de speedup sublineal. Existen varios motivos que impiden el crecimiento lineal del speedup, entre los más notorios pueden citarse las demoras introducidas por las comunicaciones y el overhead producido en el trabajo de sincronización.

El speedup y eficiencia alcanzados en los experimentos antes presentados se muestran en la Tabla 5. Observando los resultados se puede apreciar que los valores son muy buenos, siendo cercanos a los máximos teóricos. Otro resultado

Threads	**Tiempo de ejecución** (segundos)	**Desv. est.**
1	81,019	0,008
2	40,792	0,056
4	20,634	0,034
8	10,850	0,168

Tabla 4: Tiempos de ejecución para una matriz de 2048×2048.

importante es que si bien existe una degradación en la eficiencia a medida que se aumentan los threads, la misma es poco significativa. Además, si bien la cantidad de experimentos realizados no permite sacar conclusiones definitivas, parece mantenerse constante el valor de eficiencia al duplicar los threads y el tamaño del problema.

procesos	**2**	**4**	**8**
speedup-1024	1,970	3,830	6,127
eficiencia-1024	0,985	0,958	0,767
speedup-2048	1,986	3,926	7.467
eficiencia-2048	0,993	0,982	0.933

Tabla 5: Speedup y eficiencia para los sistemas de 1024 y 2048.

4. Conclusiones y trabajo futuro

El reporte presenta algunos trabajos preliminares realizados en el área de computación gráfica tendientes a avanzar en la comprensión de la algoritmia para la resolución del problema de radiosidad.

La primera conclusión es que en el trabajo se obtuvo una versión paralela utilizando el paradigma de memoria compartida para el cálculo de radiosidad. En particular, se obtuvo una versión paralela de la eliminación gaussiana utilizando OpenMP. Es importante destacar que se obtuvieron buenos valores de speedup y eficiencia, los cuales se encuentran en el entorno de los máximos teóricos.

En otro sentido, varios estudios preliminares realizados, que implican cambios más profundos en el código, no arrojaron mejoras significativas.

La conclusión principal del trabajo está referida a la gran capacidad de mejora de los tiempos de ejecución para el cálculo de radiosidad al utilizar estrategias de paralelismo.

A continuación se describen las líneas de trabajo que se están llevando adelante en la actualidad o se plantean abordar en un futuro cercano.

Un primer camino de trabajo a recorrer es extender las pruebas realizadas sobre los algoritmos de resolución de sistemas lineales utilizando la eliminación gaussiana. También resulta necesario extender el estudio para abarcar otras factorizaciones como puede ser LU.

Un problema al que nos enfrentamos durante el trabajo, es la ausencia de un conjunto de problemas de referencia que sirvan para evaluar los resultados de los métodos utilizados. Esto motiva la necesidad de presentar un conjunto de problemas a los cuales se les pueda aplicar distintas estrategias y comparar resultados.

Otra línea de trabajo a seguir consiste en migrar las ideas experimentadas de las CPU a las GPU, explotando las capacidades de procesamiento de memoria compartida que poseen.

También es de interés profundizar en la investigación sobre otras metodologías para el almacenamiento y resolución de los sistemas lineales presentes en los problemas de radiosidad.

Por último, mencionaremos como línea no abordada la búsqueda de técnicas que permitan eliminar o por los menos mitigar el costo que implica las restricciones de movimiento de objetos en las escenas al calcular radiosidad.

Referencias

[1] Sitio web con información sobre multicores de amd,. http://multicore.amd.com/es-ES/AMD-Multi-Core.aspx. Fecha de última consulta, marzo 2009.

[2] Sitio web con información sobre multicores de intel,. http://www.intel.com/espanol/business/bss/products/server/dual-core.htm. Fecha de última consulta, marzo 2009.

[3] Sitio web de gpgpu,. www.gpgpu.org. Fecha de última consulta, marzo 2009.

[4] Sitio web de openmp,. www.OpenMP.org. Fecha de última consulta, marzo 2009.

[5] Sitio web de top 500,. www.top500.org. Fecha de última consulta, marzo 2009.

[6] Sitio web del centro de supercomputación de galicia,. http://www.cesga.es. Fecha de última consulta, marzo 2009.

[7] G. M. Amdhal. Validity of the single-processor approach to achieving large scale computing capabilities. In *AFIPS*, pages 483–485, 1967.

[8] M. Cohen and J. Wallace. Radiosity and realistic image synthesis, 1993.

[9] Michael F. Cohen and Donald P. Greenberg. The hemi-cube: a radiosity solution for complex environments. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 31–40, New York, NY, USA, 1985. ACM.

[10] Andrew S. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994.

[11] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the interaction of light between diffuse surfaces. *SIGGRAPH Comput. Graph.*, 18(3):213–222, 1984.

[12] David Luebke, Mark Harris, Naga Govindaraju, Aaron Lefohn, Mike Houston, John Owens, Mark Segal, Matthew Papakipos, and Ian Buck. Gpgpu: general-purpose computation on graphics hardware. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 208, New York, NY, USA, 2006. ACM.