2010-10

Las redes de Petri en la paralelización eficiente de aplicaciones: caso de uso

Alcaraz-Mejía, Mildreth; Campos-Rodríguez, Raúl; Acosta-Lúa, Cuauhtémoc; Carranza-Sahagún, Diego U.

M. Alcaraz-Mejía, R. Campos-Rodríguez, C. Acosta-Lúa, D. Carranza-Sahagún (2010). Las redes de Petri en la paralelización eficiente de aplicaciones: caso de uso. En VI Semana Nacional de Ingeniería Electrónica. Memorias en Extenso, v.1, pp.629-636.

Las Redes De Petri En La Paralelización Eficiente De Aplicaciones: Caso De Uso

***M. Alcaraz-Mejía, *R. Campos-Rodríguez, *C. Acosta-Lúa, **D. Carranza-Sahagún**

*Departamento de Ciencias Tecnológicas, **Departamento de Ciencias Básicas,
Centro Universitario de la Ciénega, Universidad de Guadalajara, Av. Universidad No. 1115, Col. Lindavista, Ocotlán, Jalisco; México. E-mail: {mildreth.alcaraz}@cuci.udg.mx

Resumen
En este trabajo se presenta el método basado en modelos de Redes de Petri para el análisis y paralelización eficiente de aplicaciones programadas con un paradigma secuencial. Primeramente, se realiza el modelo de la aplicación secuencial. Enseguida, se analizan las partes paralelizables, y se presenta un modelo en Red de Petri de la aplicación paralelizada. A partir del modelo en Red de Petri, se realiza la verificación de la construcción del modelo y se analiza de manera informal la relación de los P-Invariantes con la paralelización del modelo. Finalmente, se realiza una comparación del tiempo de cómputo entre el paradigma secuencial y el paralelo. Se utiliza la multiplicación de matrices como caso de estudio y se reportan los resultados experimentales.

Palabras Clave: Cómputo Paralelo, Redes de Petri, Paralelización, Análisis de Complejidad, Tiempo de cómputo.

I. Introducción

Uno de los principales retos para la programación de máquinas multi-núcleo de Múltiple Instrucción Múltiple Datos (MIMD) es la decisión de qué tareas o datos son paralelizables, aunado al análisis de la eficiencia resultado de esta paralelización comparado con la programación secuencial [1]. Existen varios métodos para la implementación de la paralelización, como el OpenMP [2], Fortran [3], C++ [4], Threading Building Blocks [5], Win32 Threading API y Pthreads [6], en general, librerías de manejo de hilos. Sobre el uso de las Redes de Petri en la paralelización, existen diversos enfoques. En [7], los autores presentan una versión experimental de un traductor de modelos en Redes de Petri, al lenguaje de programación C++. Los autores incluyen comandos funcionales, operadores y tipos de datos a los modelos en Redes de Petri, lo cual resulta útil para especificar programas de cómputo. En [8], los autores proponen una extensión a la semántica de un lenguaje de programación paralela a partir de Redes de Petri de Alto nivel. Desde el punto de vista de los modelos en Redes de Petri, la solución propuesta utiliza refinamientos y operaciones de sincronización. La metodología así propuesta hereda buenas propiedades presentes en el modelo en Red de Petri. En [9], los autores presentan un método para la depuración de programas paralelos. El método incluye la simulación y verificación de programas paralelos representados mediante un lenguaje de programación gráfico, denominado P-GRADE, del cual es posible obtener Redes de Petri Coloreadas. Las redes obtenidas se pueden utilizar para verificar el comportamiento críticos del programa, como vivacidad e inter-bloqueos, mediante técnicas de Redes de Petri, como la expansión del grafo de alcanzabilidad. Todo esto se encuentra en un entorno de desarrollo integrado adecuado para los programadores. En [10], los autores proponen el uso de las Redes de Petri para la optimización y depuración de software escrito para plataformas paralelas. Resaltan los atributos gráficos de las Redes de Petri y las proponen como un lenguaje de

depuración.

En este trabajo, se presenta una primera aproximación sobre el estudio de la conversión de modelos de aplicaciones secuenciales a su equivalente modelo paralelizado basado en Redes de Petri. Se utiliza la multiplicación de matrices como caso de estudio. Se presenta el modelo en Red de Petri de la multiplicación de matrices de manera secuencial, como normalmente se resuelve en la programación estructurada. A partir del modelo en Red de Petri, se analizan las posibles conversiones secuencial-paralelas de las tareas de la aplicación. Enseguida, se obtiene el modelo en paralelo de la aplicación. Después, se analizan las posibles paralelizaciones a través del análisis de su estructura y los P-invariantes que presenta el modelo, que junto con un análisis del desempeño, permiten realizar la mejor selección de paralelización posible.

El trabajo se organiza de la siguiente manera. La sección II presenta el desarrollo del trabajo presentado. La sección III presenta los resultados de la implementación en secuencia y en paralelo del caso de uso y su análisis comparativo. Finalmente, se presentan conclusiones y trabajo futuro.

II. Desarrollo

Las Redes de Petri (RP) es una metodología formal de modelado de sistemas. Sus principales características son la representación dinámica gráfica y sus fuertes bases matemáticas que permiten el análisis formal, así como obtener información sobre el comportamiento dinámico del sistema modelado [11].

De manera informal, una Red de Petri se define como un par (G, M_0), donde G define la estructura de la red con una 4-tupla: P es el conjunto de lugares, T es el conjunto de transiciones, $I: P \times T \rightarrow Z^+$ es una función que define la incidencia de entrada a las transiciones, y $O: P \times T \rightarrow Z^+$ es una función que define la incidencia de salida de las transiciones. Dada la naturaleza de estas funciones I, O se representan con matrices del tipo $I_{|P|\times|T|}$ y $O_{|P|\times|T|}$. M_0 es una función que mapea a cada lugar en la red un entero positivo Z^+ que representa el número de marcas que reside en cada lugar, comúnmente denominado Marcado ó Estado de la red.

Comúnmente, los lugares se representan con círculos, las transiciones con rectángulos, la función de entrada con arcos del lugar a la transición, y la función de salida con arcos de la transición al lugar indicado. Las marcas se representan con pequeños círculos negros dentro de los círculos que representan los lugares.

Para este trabajo, se considera el caso de la multiplicación matricial generalizada de dos matrices $C_{m\times n} = A_{m\times p} B_{p\times n}$ que se define formalmente de la siguiente manera:

$$c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj} \tag{1}$$

Para $\forall i, j$ *tal que* $i \in [1..m]$ *y* $j \in [1..n]$.

donde,

$$c_{11} = a_{11}b_{11} + \cdots + a_{1p}b_{p1},$$
$$\vdots$$
$$c_{1n} = a_{11}b_{1n} + \cdots + a_{1p}b_{pn}$$
$$\vdots$$
$$c_{m1} = a_{m1}b_{11} + \cdots + a_{mp}b_{p1}$$
$$\vdots$$
$$c_{mn} = a_{m1}b_{1n} + \cdots + a_{mp}b_{pn}$$

Explícitamente, se tiene que:

$$\begin{bmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{m1} & \cdots & c_{mn} \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{1p} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mp} \end{bmatrix} \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{p1} & \cdots & b_{pn} \end{bmatrix}, \tag{2}$$

La manera común de programar esta multiplicación en un paradigma de programación secuencial, es utilizando tres estructuras `for`

 ISBN 978-607-477-363-7

```
//c[i,j]=0 para todos los valores de i,j
for(int i=0;i<m;i++)
     for(int j=0; j<n; j++)
          for(int k=0;k<p; k++)
               c[i,j] += a[i,k]* b[k,j];
```

Fig. 1. Código genérico de la multiplicación en matrices de manera secuencial.

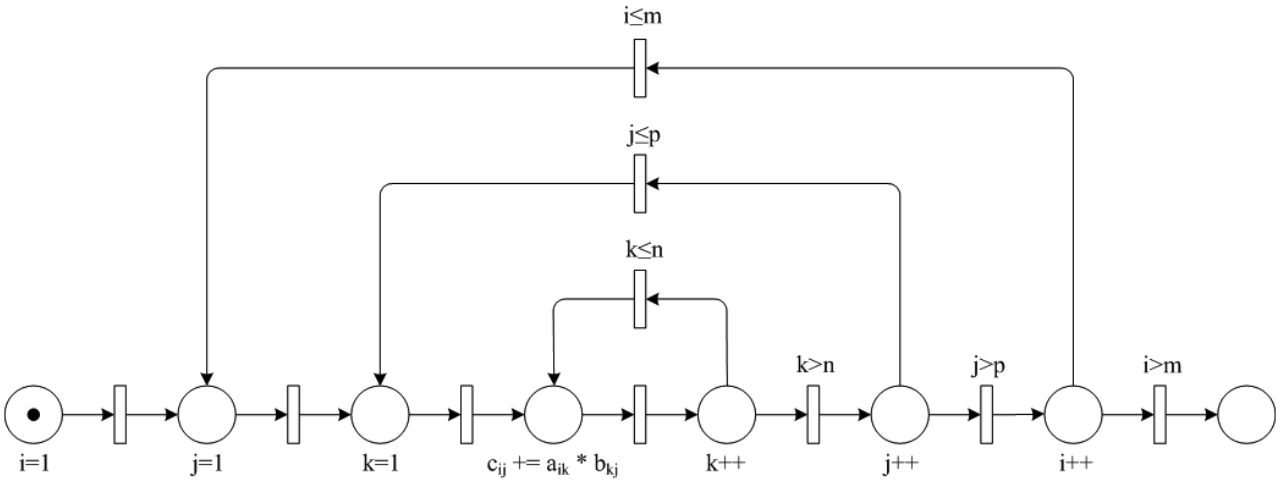

Fig. 2. Red de Petri que modela el código de la multiplicación de matrices con `for` anidados.

anidadas. En la Fig. 1 se muestra el código generalizado en C de esta aplicación.

Una forma de modelar esta estructura de programación anidada secuencializada es utilizando las Redes de Petri. En la Fig. 2 se muestra la propuesta de modelado de esta aplicación en Red de Petri. En este caso, los lugares representan operaciones y las transiciones representan condiciones, donde $i, j, k, m, n, p \in Z^+$ y m, n, p están definidos.

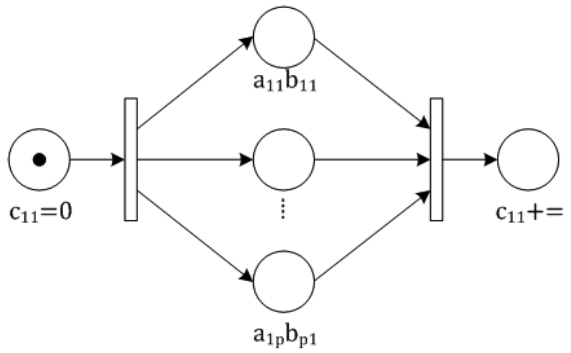

Fig. 3. Suma de multiplicaciones en paralelo

Se puede observar que la Red de Petri es una máquina de estados que conectando el último lugar con el primer lugar de la red a través de una transición, la red tiene un solo p-semiflujo, por lo tanto, es una máquina de estados, es viva y acotada, y tiene 4 t-semiflujos. Para profundizar sobre las propiedades de la Red de Petri, se refiere al lector a [12],[13].

Se observa que la red en cuestión tiene ciclos que pueden inducir a que ésta sea infinita. En este caso en particular, debido a que las transiciones representan condiciones que se deben de cumplir para realizar el evento, y dado que los ciclos están acotados por los índices m, n, p, el número máximo de ciclos está dado por $m \times n \times p$. Bajo estas condiciones, la red se puede "desdoblar" convirtiendo los ciclos a un número finito de m, n, p pasos, obteniendo una red totalmente secuencial y finita.

Este análisis concuerda con el análisis de complejidad que se tiene en el código que se presenta en la Fig. 1, donde éste queda en el

 ISBN 978-607-477-363-7

orden de:

$$O(mnp), \quad (7)$$

es decir, el algoritmo es de orden cúbico, como es bien conocido para la multiplicación de matrices en un paradigma de programación secuencial.

Cuando se desea cambiar una aplicación de un paradigma secuencial a uno paralelo, este cambio no es directo, puesto que debe haber un análisis para identificar si la aplicación es paralelizable,

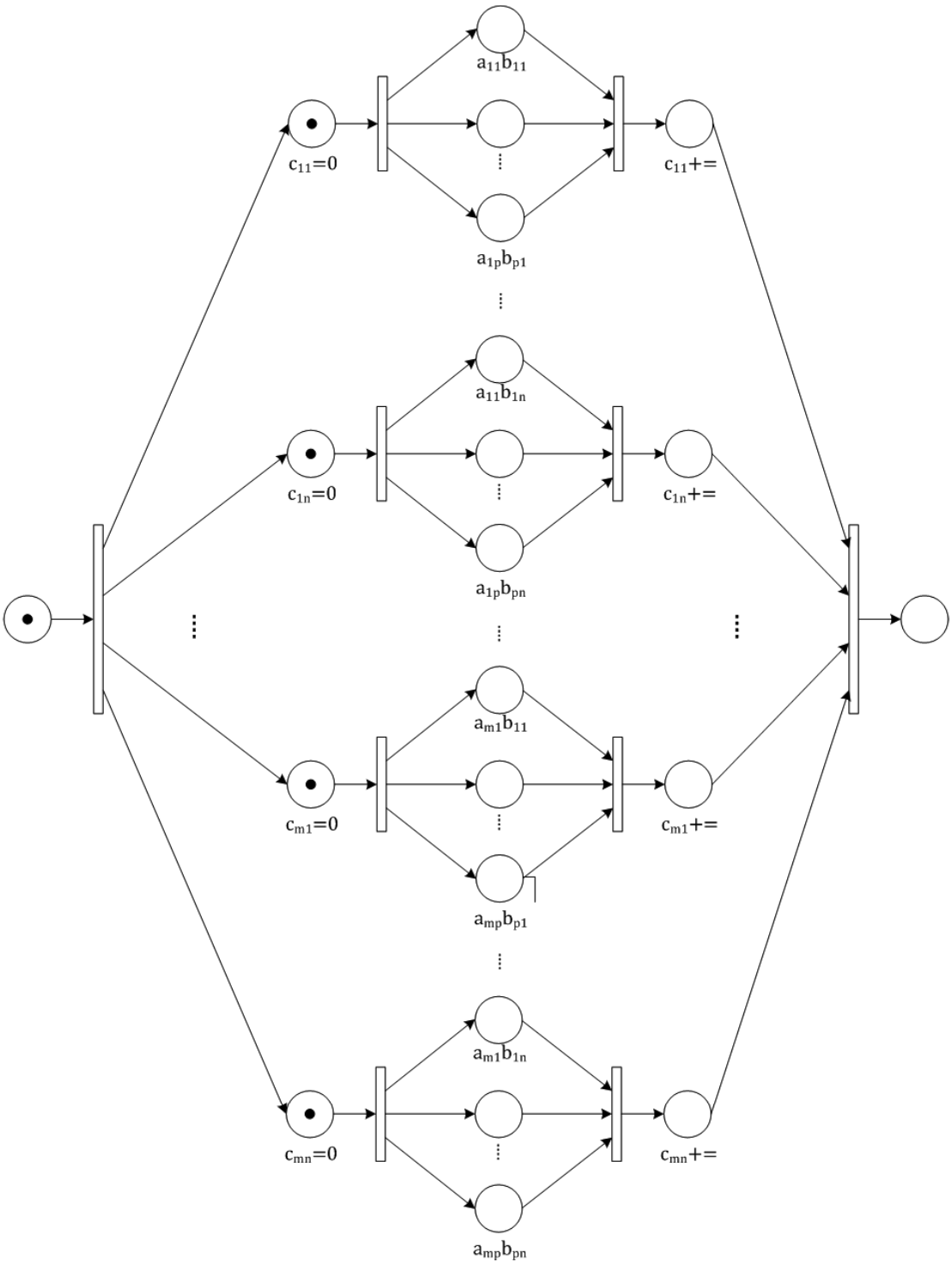

Fig. 4. Red de Petri que modela la multiplicación de matrices en paralelo.

 ISBN 978-607-477-363-7

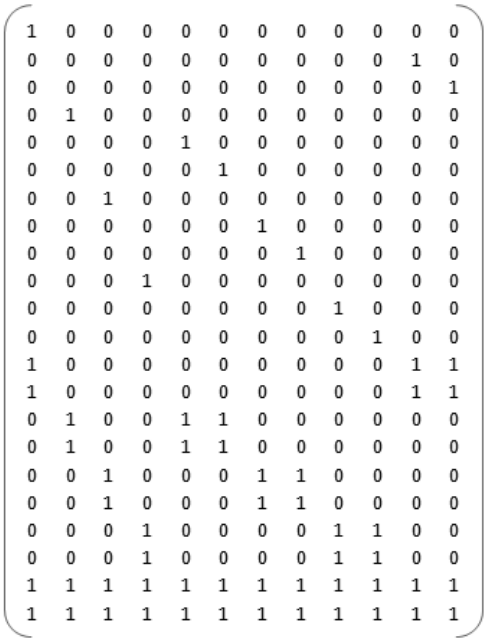

Fig. 5. P-semiflujos de la red

en qué partes es paralelizable, y cuál paralelización es la más conveniente en términos de eficiencia.

En este caso, se realiza el análisis sobre la red de la Fig. 2, iniciando por las operaciones que se encuentran en el ciclo más interno de la red, es decir, considerando $c_{ij} += a_{ik}b_{kj}$. Puesto que la operación de $a_{ik}b_{kj}$, no depende de ningún valor calculado previamente como se observa en la ecuación (2), entonces, para $i = j = 1$, y para todo valor de $1 \leq k \leq p$, como en la ecuación (3) es posible paralelizar como se muestra en la Red de Petri en la Fig. 3.

Si repetimos el mismo esquema para todo valor de i, j tal que $1 \leq i \leq m, 1 \leq j \leq n$, conociendo que para cada término c_{ij} los cálculos son independientes entre ellos, entonces, se obtiene un modelo en Red de Petri como el que se observa en la Fig. 4.

Independientemente de los valores que tomen m, n, p, esta red tiene un solo t-invariante semipositivo que incluye a todas las transiciones de la red, por lo tanto, la red corresponde a un Grafo Marcado [12],[13]. Para una matriz A de 2×3 y una B de 3×2, puesto que C es de 2×2, entonces la red tiene 12 p-invariantes semipositivos, los cuales se muestran en la Fig. 5. Para esta aplicación, se puede observar que la cantidad de p-invariantes semipositivos proporcionan la cota superior de posibles bloques paralelizables. Es decir, hay 12 operaciones básicas $a_{ik}b_{kj}$ en total que se pueden hacer en paralelo, es decir 12 tareas. El tiempo de cálculo total considerando que cada operación básica se

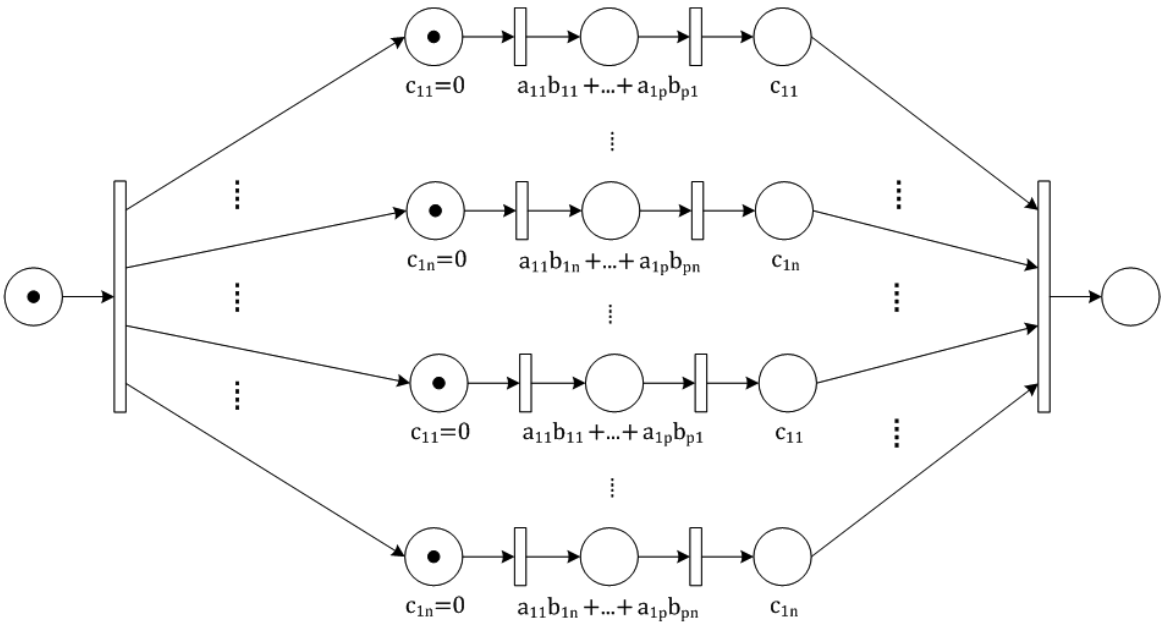

Fig. 6. Red de Petri paralelizada para toda c_{ij}.

 ISBN 978-607-477-363-7

tarda un tiempo t, quedaría acotado por:

Si $mnpModQ = 0$ entonces

$$\frac{mnp}{Q} \cdot t, \tag{8}$$

De otro modo,

$$\left(\frac{mnp}{Q} + 1\right) \cdot t, \tag{9}$$

donde m, n, p son los índices de las matrices, Q es el número de procesadores disponibles, y considerando que cuando se cumple que $mnpModQ > 0$ se asigna un cálculo a cada procesador hasta que se terminen los cálculos.

Observe que el número de cálculos totales que tiene este problema es exactamente el mismo para en la solución secuencial que en la paralela. Esto es, el paralelismo no disminuye el número total de cálculos que se deben realizar, solo disminuye el tiempo de cómputo. Para el ejemplo, suponiendo que Q=$m \times n \times p$ procesadores, es decir, un procesador para cada tarea, el tiempo de cómputo es t.

Observe que otra forma de paralelizar sugerida por la red tipo Grafo Marcado es por términos c_{ij} donde cada tarea debería realizar la sumatoria representada en la ecuación (1), para un i, j fijo. La red que representa esta paralelización se muestra en la Fig. 6.

En este caso, se observa que se tienen mn tareas a paralelizar en Q procesadores, si tenemos que $Q = m \times n$, entonces el tiempo de cómputo sería pt. Aunque el análisis de la paralelización realizada como en la Fig. 4, comparado con el análisis de la paralelización como en la Fig. 6 sugiere que es mejor la primera que la segunda, no se debe olvidar que el tiempo requerido para la separación del problema, como el de conjunción de resultados (como en este caso la sumatoria), debe considerarse en el cálculo del tiempo total.

III. Resultados

Para el caso de uso se implementaron los algoritmos en secuencia y en paralelo de tareas para el caso de uso de la multiplicación de matrices en C++. Se utilizó un procesador Intel Core Quad Q8400 2.66GHz con 4Gbytes de RAM con Windows 7 Ultimate de 64 bits, para todos los casos.

Se hicieron algunos experimentos para calcular los tiempos de cómputo para 5 diferentes casos por tamaños de matrices, donde el llenado de cada celda a_{ik}, b_{kj} se realizó aleatoriamente con números enteros del 0 al 100.

Caso 1. Tamaños: $A_{2\times3}, B_{3\times2}$ y $C_{2\times2}$.

Caso 2. Tamaños: $A_{20\times15}, B_{15\times10}$ y $C_{20\times10}$.

Caso 3. Tamaños: $A_{60\times60}, B_{60\times60}$ y $C_{60\times60}$.

Caso 4. Tamaños: $A_{100\times50}, B_{50\times100}$ y $C_{100\times100}$.

Caso 5. Tamaños: $A_{1000\times800}, B_{800\times500}$ y $C_{1000\times500}$.

Los resultados del tiempo de cómputo para cada caso en cada variante se resumen en la Tabla 1.

Caso	*T Secuencia*	*T paralelo*
1	63	23
2	2,541	26
3	81,028	650
4	193,284	64,676
5	92’142,811	2’504,106

Tabla 1. Resumen de resultados prácticos

En la Tabla 1, los tiempos *T Secuencia* y *T Paralelo* están dados en ticks, donde $tick = 100nSeg$.

IV. Conclusiones

Actualmente, con el avance tecnológico que existe en las arquitecturas multi núcleo, se ha incrementado la necesidad de metodologías que apoyen en el análisis de aplicaciones desarrolladas bajo un paradigma secuencial, para su posible paralelización. En este trabajo utilizamos las redes de Petri para analizar las partes paralelizables de una aplicación en particular, se realizó la implementación, y se obtuvo una considerable disminución en el tiempo de cómputo entre la versión secuencial y

 ISBN 978-607-477-363-7

la paralela. Se puede observar que las redes de Petri permiten una buena representación de la paralelización, y sus bases matemáticas permiten verificar propiedades importantes. En este sentido, obtener una metodología para el análisis de programas secuenciales, la obtención de su modelo, si es posible la paralelización de la aplicación, y su análisis de eficiencia es un tema de estudio a seguir.

Referencias

[1]. S. Akhter, J. Roberts, Multi-Core Programming, Abril 2006, Intel Press, OR, Estados Unidos de América. ISBN 0-9764832-4-6.

[2]. The OpenMP API specification for parallel programming, http://openmp.org/wp/, Última actualización en June 7, 2010.

[3]. S.J. Chapman, Fortran 95/2003 for Scientists and Engineers, 3rd edition, 2007, McGraw-Hill. ISBN 978-0-07-319157-7.

[4]. Visual C++ 2010 And The Parallel Patterns Library, http://msdn.microsoft.com/en-us/magazine/dd434652.aspx, Febrero 2009.

[5]. Intel® Threading Building Blocks 3.0, www.threadingbuildingblocks.org, Mayo 2010.

[6]. R. H. Carver, K.C. Tai, Modern multithreading: implementing, testing, and debugging multithreaded Java and C++/Pthreads/Win32 Programs, John Wiley & Sons, 2006, Estados Unidos de América, ISBN-10: 0-471-72504-8.

[7]. E. A. Golenkov, A. S. Sokolov, G. V. Tarasov and D. I. Kharitonov. "Experimental Version of Parallel Programs Translator from Petri Nets to C++", Parallel Computing Technologies, Lecture Notes in Computer Science, 2001, Volume 2127, pp. 226-231.

[8]. H. Klaudel, Compositional high-level Petri net semantics of a parallel programming language with procedures, Science of Computer Programming, Elsevier Science, Volume 41, Issue 3, Nov. 2001, pp. 195-240.

[9]. R. Lovas and B. Vécsei, Integration of Formal Verification and Debugging Methods in P-GRADE Environment, Distributed and Parallel Systems, The Kluwer International Series in Engineering and Computer Science, 2005, Volume 777, Part III, pp. 83-92.

[10]. D. I. Kharitonov, G. V. Tarasov. "Towards Petri nets application in parallel program debugging". The 6th Asian Computational Fluid Dynamics Conference, Taiwan, Oct. 2005.

[11]. M. Silva, Las redes de Petri: En la automática y la informática, Editorial AC, 1985, ISBN-10: 8472880451.

[12]. J. Desel and J. Esparza, Free Choice Petri Nets, Cambridge Tracts in Theoretical Computer Science, Cambridge University Press; Nueva Edición, Septiembre 8, 2005, ISBN-10: 0521019451.

[13]. T. Murata, "Petri Nets: Properties, Analysis and Applications," Proceedings of the IEEE, Vol. 77, No. 4, pp. 541–580, 1989.

V. Autores

Dra. Mildreth Isadora Alcaraz Mejía es Ingeniero en Sistemas Computacionales por el Instituto Tecnológico de Colima (2001). Obtuvo los títulos de Maestro (2004) y Doctor (2007) en Ciencias con especialidad en Ingeniería Eléctrica en el CINVESTAV Unidad Guadalajara. Actualmente, se dedica a la aplicación y desarrollo de técnicas de control embebidas, al diseño de metodologías de programación, y al estudio de sistemas híbridos.

Dr. Raúl Campos Rodríguez es Ingeniero en Computación, Centro Universitario de la Ciénega, UDG (2000). Estudió la Maestría (2002) y Doctorado (2007) en Ciencias en Ingeniería Eléctrica en el CINVESTAV Guadalajara. Trabaja activamente en el diseño de algoritmos para sistemas de eventos discretos y en el diseño de Microsistemas, o MEMS, específicamente en mecanismos basados en fuerza electrostática y expansión térmica.

Dr. Cuauhtémoc Acosta Lúa es Ingeniero en Electrónica por el Instituto Tecnológico de

Morelia (2001). Obtuvo los títulos de Maestro (2003) y Doctor (2007) en Ciencias con especialidad en Ingeniería Eléctrica en el CINVESTAV Unidad Guadalajara. Participó en estancias en el INSA Lyon, Francia y Centro de Investigación DEWS en L'Aquila Italia. Tiene un Posdoctorado en conjunto del Centro de Investigación DEWS de L'Aquila, Italia y el Centro de Investigación y Aplicación de la Ford Motor Company. Actualmente se dedica al desarrollo de técnicas de control no lineal para vehículos, así como observadores no lineales para distintos subsistemas de los mismos y estudia el comportamiento de Redes de Petri Híbridas.

M. en C. Diego Ulises Carranza Sahagún es Ingeniero en Computación por la Universidad de Guadalajara. Obtuvo el grado de Maestría en Computación Aplicada, (mención Programación) en la Universidad Marta Abreu de las Villas, Cuba. Actualmente se desempeña en el área de programación de sistemas.