Implementación del Método de Descomposición en Dominios en AZTRAN

Julian A. Duran Gonzalez, Edmundo del Valle Gallegos
Instituto Politécnico Nacional, Escuela Superior de Física y Matemáticas
Av. Luis Enrique Erro s/n, Unidad Profesional Adolfo López Mateos, Zacatenco, Delegación Gustavo A. Madero, C.P. 07738, Ciudad de México; México.
redfield1290@gmail.com; edmundo.delvalle@gmail.com

Armando M. Gómez Torres
Instituto Nacional de Investigaciones Nucleares
Carretera México Toluca-La Marquesa s/n, Ocoyoacac, Estado de México, C.P. 52750
armando.gomez@inin.gob.mx

Resumen

En el presente trabajo se describe la implementación del método de descomposición en dominios al código de transporte de neutrones AZTRAN (**AZ**tlan **TRAN**sport). El código es una herramienta desarrollada para la Plataforma **AZTLAN**, el cual está escrito en Fortran 90 para el análisis neutrónico de reactores nucleares. AZTRAN resuelve numéricamente la ecuación de transporte con dependencia en tiempo con geometría cartesiana en ordenadas discretas. La arquitectura de las computadoras en la actualidad permite que los códigos se ejecuten en paralelo usando sus componentes como los procesadores o los GPUs con el fin de reducir el tiempo de cómputo. Es debido a esto que se ha hecho un esfuerzo en paralelizar códigos que resuelven modelos físicos como la ecuación de calor o difusión, en los cuales se ha podido implementar el método de descomposición en dominios. El método consiste en descomponer el dominio espacial en subdominios, los cuales van a ser distribuidos en procesadores con el fin de que cada proceso resuelva en simultáneo el subdominio y al final de aproximar el flujo en cada subdominio se actualizan las fronteras. Este método fue implementado en la versión bidimensional estacionaria del código AZTRAN utilizando la interfaz de paso de mensaje (MPI), en la cual se simuló el Benchmark C5G7-2D en estado estacionario, el cual presentó resultados alentadores que motivan a realizar una versión completamente paralelizada tridimensional.

1. INTRODUCCIÓN

En el proyecto AZTLAN Platform [1][2], el cual está en desarrollo, las herramientas para análisis el neutrónico están actualizándose constantemente. Tal es el caso del código AZTRAN [3][4], que resuelve numéricamente la ecuación de transporte (1) con dependencia en tiempo y en geometría cartesiana, discretizando sus 7 variables independientes con los siguientes métodos:

variable espacial $(\vec{r})$: método nodal RTN-0 (Raviart-Thomas-Nédélec)
variable energía (E): método de multigrupos
variable ángulo $(\hat{\Omega})$: método de ordenadas discretas S_N
variable tiempo (t): método θ

$$\frac{1}{v(E)}\frac{\partial}{\partial t}\psi(\vec{r},\hat{\Omega},E,t)+\hat{\Omega}\bullet\vec{\nabla}\psi(\vec{r},\hat{\Omega},E,t)+\Sigma_t(\vec{r},E,t)\psi(\vec{r},\hat{\Omega},E,t)=S(\vec{r},\hat{\Omega},E,t) \qquad (1)$$

donde $\psi(\vec{r},\hat{\Omega},E,t)$ es el flujo angular, $v(E)$ es la velocidad de los neutrones, $\Sigma_t(\vec{r},E,t)$ es la sección eficaz macroscópica total, y $S(\vec{r},\hat{\Omega},E,t)$ es la fuente de neutrones por precursores de neutrones retardados, fisiones y dispersiones. Para resolver la ecuación, el flujo neutrónico es obtenido usando el método de iteración de fuente y el factor de multiplicación efectiva se resuelve con el método de las potencias.

Actualmente las computadoras tienen una capacidad de cómputo considerable, por lo que hay un interés en las simulaciones numéricas en paralelizar sus códigos para reducir considerablemente el tiempo de cómputo. De hecho, hay casos de códigos neutrónicos que utilizan la técnica de ordenadas discretas como por ejemplo DENOVO [5], PARTISN [6], y PENTRAN [7] que han sido paralelizados. Los desarrollos más comunes para paralelizar las variables espaciales son el método Koch-Baker-Alcouffe (KBA) [8] y el método de Descomposición en Dominios (DDM) [9][10]. Para este trabajo se optó por implementar el segundo.

En AZTRAN se implementó una versión paralelizada por grupo de energía [11], con el problema de que la versión limita los procesos al número de grupos de energía del problema. Para poder hacer mayor uso de los recursos computacionales se implementó el método DDM en la versión bidimensional estacionaria del código mediante la utilización de la herramienta de interfaz de paso de mensajes (MPI) [12], con lo cual se obtuvo una reducción de tiempo admisible.

2. MÉTODO DE DESCOMPOSICIÓN EN DOMINIOS

En los últimos años, se ha prestado mucha atención en los métodos de descomposición en dominios ya que permiten obtener la solución de ecuaciones diferenciales parciales que se basan en una partición del dominio del problema físico. En general el método consiste en dividir el dominio en pequeños subdominios los cuales pueden ser asignados a diferentes procesadores e ir iterando para coordinar la solución entre subdominios adyacentes los cuales van a necesitar comunicarse. Dado que los subdominios se pueden manejar independientemente, estos métodos son muy atractivos para la computación en paralelo.

Hay dos formas de trabajar la descomposición en dominios: con particiones con traslape y sin traslape. En el primero cada subdominio puede traslaparse con sus vecinos y en el segundo los subdominios sólo se intersecan en las fronteras; en la Figura 1 se muestra los dominios.

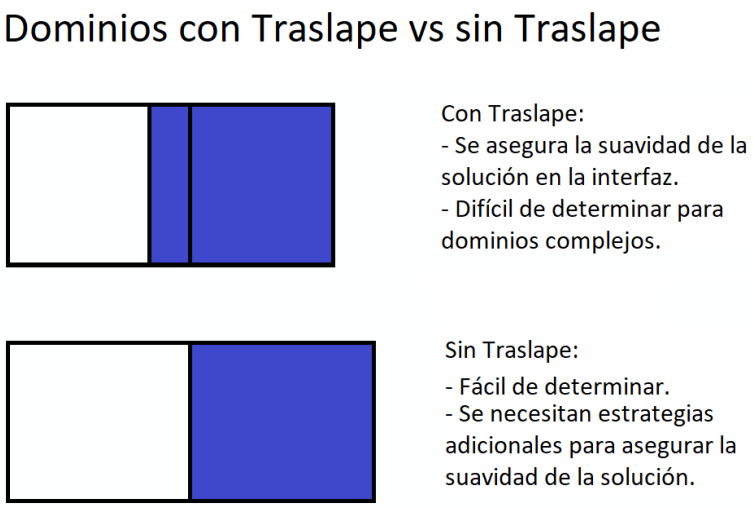

Figura 1. Dominios con traslape vs sin traslape

En la Figura 2 se muestra un ejemplo de nuestro caso de interés.

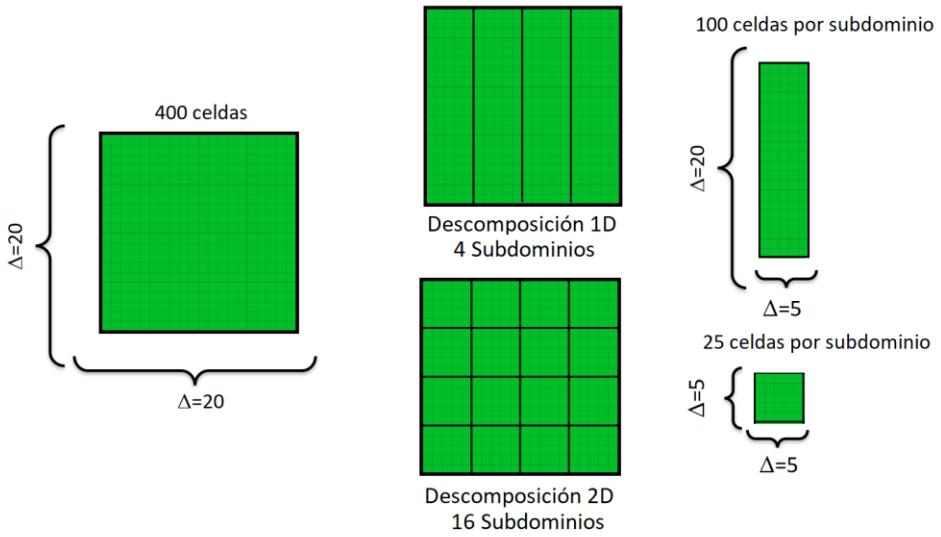

Figura 2. Tipo de descomposición problema bidimensional

2.1. Implementación del DDM en AZTRAN

Para la implementación en el código AZTRAN se seleccionó una descomposición 2D con dominio sin traslape. Además, se dispuso que los subdominios sean balanceados, esto quiere decir que cada proceso tiene el mismo número de celdas. Si el problema no presentara un dominio balanceado se tendría que hacer un refinamiento interno en el archivo de entrada lo que significa aumentar la partición espacial, todo esto poder balancearlo. En la Figura 3 se presenta la forma en que se distribuye los subdominios en 16 procesos en AZTRAN.

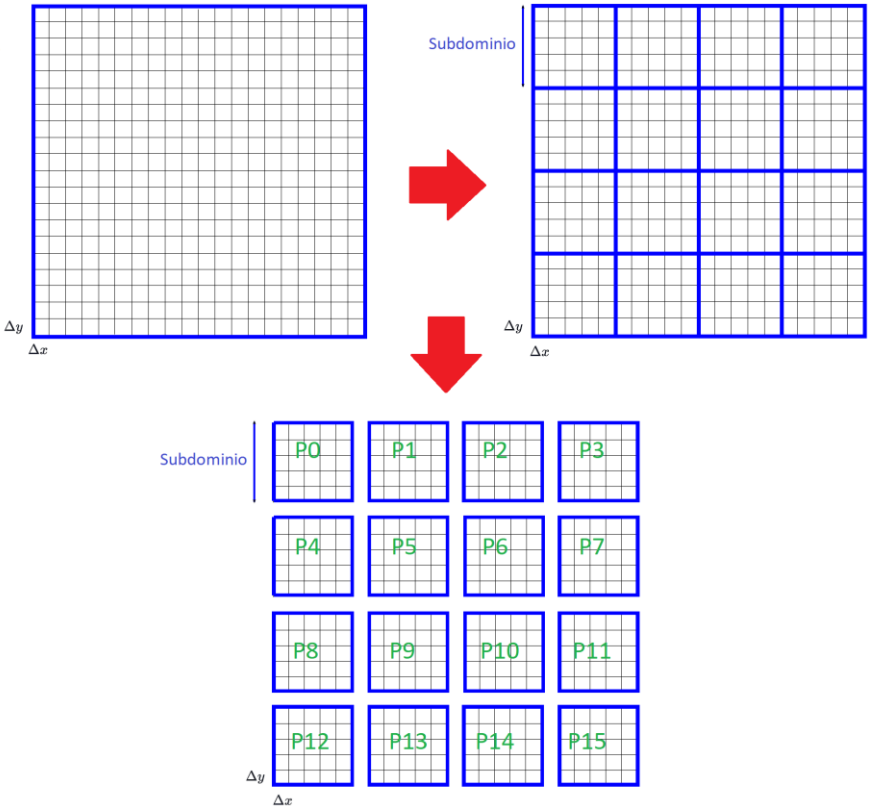

Figura 3. Descomposición de Dominio en AZTRAN

Para la implementación del DDM en AZTRAN se utilizó la interfaz de paso de mensajes (MPI). Para empezar, es necesario la creación de una topología cartesiana virtual que permite la partición del dominio en los procesadores, la biblioteca MPI proporciona las herramientas para la creación de una topología. La topología cartesiana es generada con la función MPI_CART_CREATE() la cual crea una nueva comunicación con la topología dada; la función MPI_CART_COORDS() devuelve las coordenadas del comunicador cartesiano con el cual se puede construir el dominio de cada subdominio y finalmente con la función MPI_CART_SHIFT() se puede conocer los subdominios vecinos de cada procesador.

Ya con los subdominios asignados en los procesos, el problema del flujo neutrónico es resuelto con el método de iteración de fuente y al resolver los subdominios en paralelo se procede a actualizar las fronteras de los subdominios usando la función MPI_SENDRECV() con la cual en una sola llamada se mandan los valores de la frontera de un subdominio y lo recibe el proceso que contiene al subdominio vecino. Cabe resaltar que la comunicación depende de la dirección angular. En la Figura 4 se muestra cómo se comunican los subdominios para cada dirección angular en un problema bidimensional donde $\hat{\Omega} = (\mu, \eta)$.

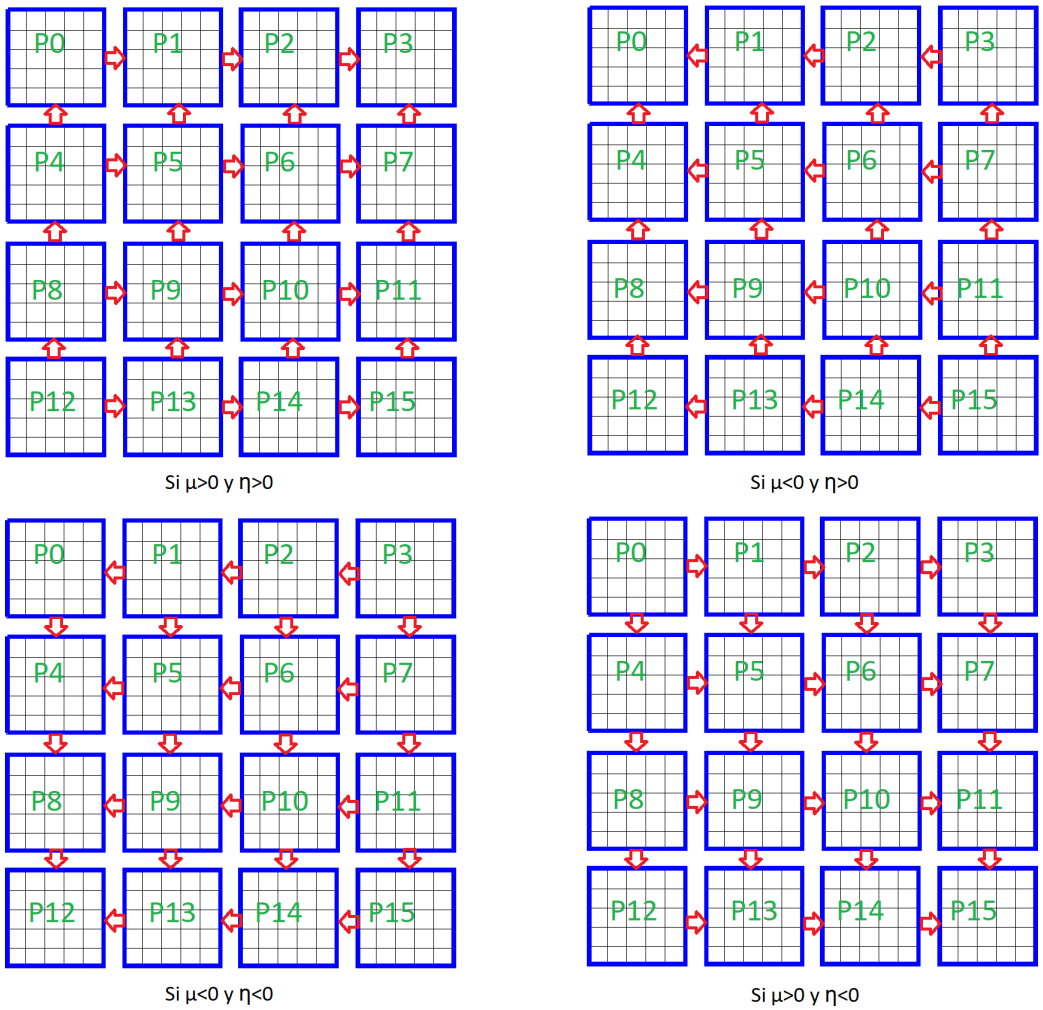

Figura 4. Comunicación de subdominios

3. RESULTADOS

Para la verificación se consideró el Benchmark C5G7-2D el cual es un problema adecuado para esta metodología debido a su complejidad por tamaño, diseño y heterogeneidad. Este problema de 7 grupos de energía es modelado con un cuarto de núcleo con simetría, las dimensiones generales son 64.26 x 64.26 cm, cada ensamble es de 21.42 x 21.42 cm, las condiciones de frontera aplicadas son reflexivas en la parte superior e izquierda, mientras en la derecha e inferior se aplican condiciones de vacío. Cada ensamble de combustible está compuesto por una celda de 17x17. En la Figura 5 se muestra la configuración del problema y una descripción detallada del Benchmark junto a sus secciones eficaces pueden ser encontradas en la referencia [13].

El equipo utilizado para realizar las simulaciones cuenta con una familia de procesadores Intel® Xeon® CPU E5-2699 v3 @2.30GHz x 72.

En la Tabla I se muestran los resultados obtenidos por AZTRAN con una tolerancia de 1.0E-08 para las iteraciones internas y 1.0E-07 para las externas, los cuales se comparan con PARTISN en el trabajo [14].

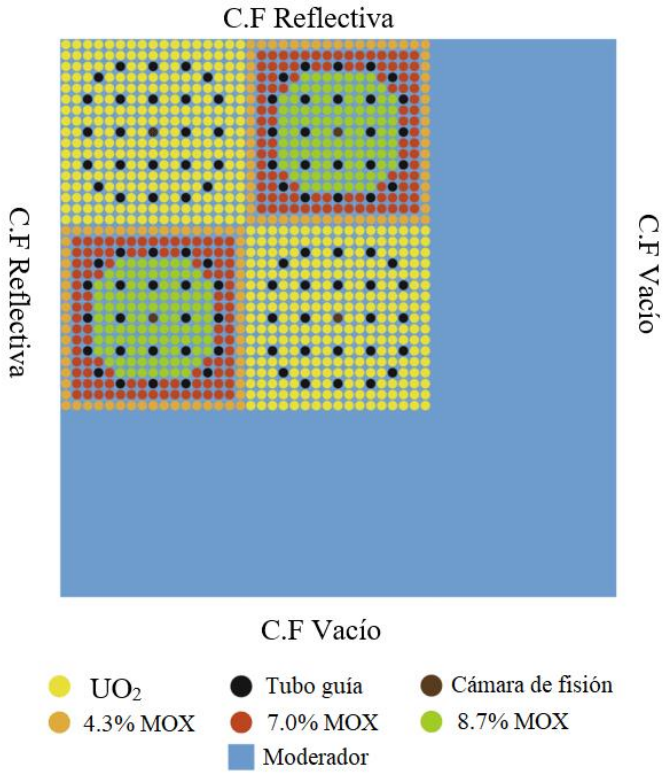

Figura 5. Modelo Benchmark C5G7-2D

Tabla I. Factores de multiplicación efectiva C5G7-2D

Ordenada Discreta	k_{eff} (AZTRAN)	k_{eff} (PARTISN)	Diferencia pcm
S_2	1.18654	---	
S_4	1.18473	1.18459	-12 pcm
S_6	1.18480	1.18468	-10 pcm
S_8	1.18501	1.18477	-20 pcm
S_{12}	1.18556	1.18556	0 pcm
S_{16}	1.18598	1.18599	1 pcm

$$pcm = \frac{k_{ref} - k_{AZTRAN}}{k_{ref}} \times 10^5 \tag{2}$$

Para resolver el problema con DDM se les asignó a los procesadores diferentes configuraciones de subdominios para poder observar el comportamiento de cómo es acelerado el problema, en la Figura 6 se muestran las configuraciones propuestas.

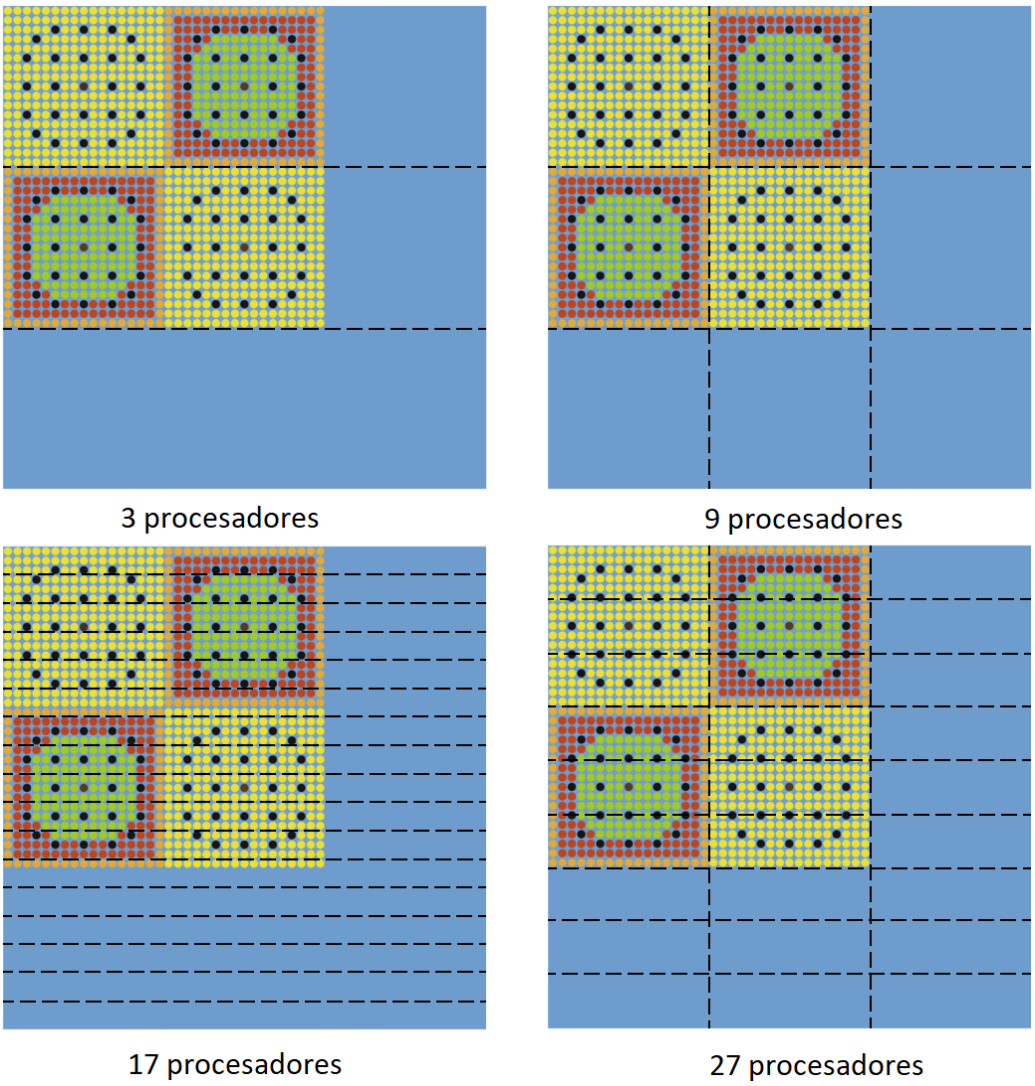

Figura 6. Configuración de subdominios propuestos

Aplicando estas configuraciones se obtuvieron la Tabla II y III que muestran los resultados de los tiempos de cómputo, y así como el número de iteraciones internas totales.

Tabla II. Tiempos de cómputo sin refinamiento

Número de procesadores	S_2	S_4	S_6	S_8	S_{12}	S_{16}
1	441 s	1626 s	5091 s	5816 s	19110 s	35649 s
3	334 s	1066 s	3713 s	3813 s	13733 s	22676 s
9	258 s	722 s	2313 s	2473 s	10556 s	13317 s
17	359 s	996 s	2816 s	3326 s	13153 s	18697 s
27	431 s	1235 s	3385 s	4058 s	11958 s	25300 s

Tabla III. Tiempos de iteraciones internas totales sin refinamiento

Número de procesadores	S_2	S_4	S_6	S_8	S_{12}	S_{16}
1	14946	15201	15163	15183	15146	15140
3	24059	24377	24388	24394	24404	24069
9	25572	26197	25841	26194	25874	25806
17	30142	29967	30001	29387	30047	29538
27	28193	28492	28493	28495	28500	28680

En las Tablas IV y V se muestran resultados, los cuales se obtuvieron aplicando un incremento en el refinamiento espacial en el archivo de entrada.

Tabla IV. Tiempos de cómputo con un refinamiento (2x2)

Número de procesadores	S_2	S_4	S_6	S_8	S_{12}	S_{16}
1	2545 s	6884 s	16050 s	26434 s	54957 s	130127 s
3	1678 s	4544 s	15989 s	19783 s	33533 s	88810 s
9	1096 s	2973 s	9826 s	13645 s	32186 s	52444 s
17	1448 s	4014 s	11976 s	17497 s	39445 s	71441 s
27	1783 s	4950 s	12674 s	21665 s	47494 s	79774 s

Tabla V. Tiempos de iteraciones internas totales con un refinamiento(2x2)

Número de procesadores	S_2	S_4	S_6	S_8	S_{12}	S_{16}
1	14630	15154	15180	15123	15132	14578
3	23794	24379	24397	23467	24407	24414
9	25657	26193	26196	26193	25878	25890
17	30125	30017	30044	30038	30060	30072
27	28319	28511	27872	28512	28516	28517

En las Tablas II y IV se observa que hay un ahorro de tiempo plausible respecto al cálculo con un procesador, y además al aumentar el número de procesadores no lleva necesariamente a mayor aceleración y eso se debe a que existe mayor comunicación.

Finalmente, en las Tablas III y V se puede notar que el número de iteraciones internas totales se incrementa al realizar una descomposición en dominios, y esto es consecuencia de que al tener mayores subdominios se necesitan más iteraciones para converger, ya que al inicio del proceso iterativo se empieza con un valor inicial pero solo unos cuantos subdominios son resueltos bien ya que son los que están cerca de la frontera donde se conoce el valor del flujo, así que se necesitan iteraciones extras para que se actualicen los otros subdominios alejados de las fronteras.

Para poder cuantificar la aceleración del método se define el "speedup" en la ecuación (3), el cual está dado como el cociente del tiempo obtenido en serie entre el tiempo resultante en paralelo.

$$S_p = \frac{T_s}{T_p} \tag{3}$$

En la Figura 7 se muestra el "speedup" obtenido para cada refinamiento y aproximación angular, en el cual se puede observar que el mayor ahorro se obtiene con la configuración con 9 procesadores, ya que se tiene una distribución ideal, además ya se ha documentado que la descomposición 2D tiene mayor aceleración que la 1D.

También se aprecia que al aumentar la aproximación angular se tiene un mayor "speedup". Esto se debe a que antes de actualizar las fronteras de los subdominios, el algoritmo puede trabajar más tiempo en paralelo antes de que exista la comunicación entre procesadores.

Finalmente se puede apreciar que en los refinamientos 2x2 sus comportamientos son erráticos y es probable que al aumentar el mallado espacial, se debe seleccionar otra configuración de subdominios con la cual se pueda obtener mejor "speedup".

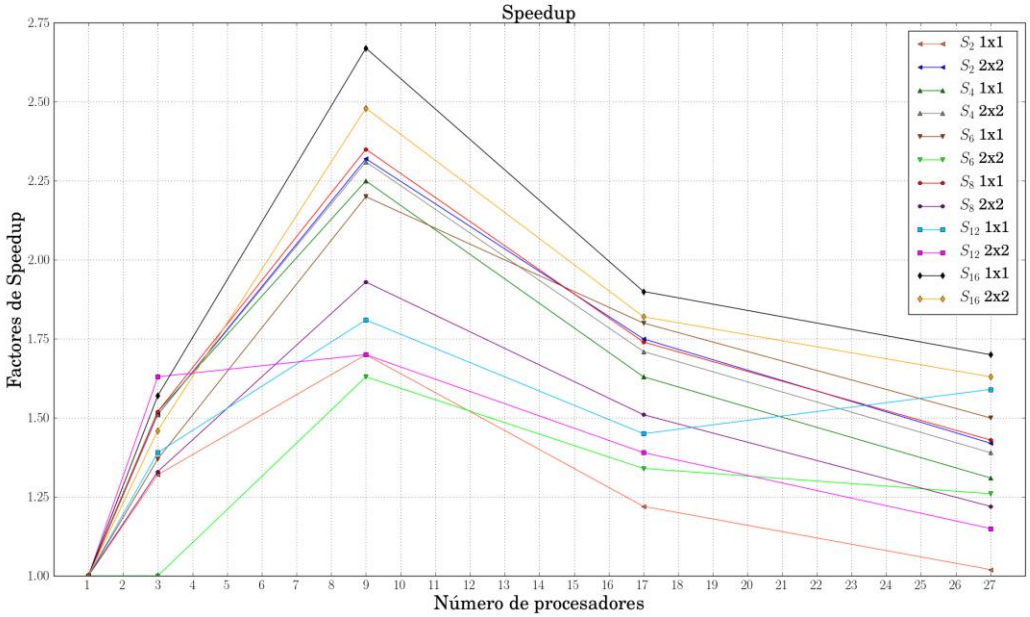

Figura 7. Speedup obtenido para cada configuración

4. CONCLUSIONES

En el presente trabajo se presentó la implementación del método de descomposición en dominios al código AZTRAN, obteniendo resultados alentadores. Al realizar las pruebas se encontró que el método existe mayor "speedup" al aumentar la aproximación angular, además el método está muy ligado a la discretización espacial con lo cual al aumentar el dominio espacial se puede hacer uso de mayores recursos computacionales, aunque como se muestra en el trabajo se necesita realizar una descomposición 2D balanceada para obtener el mayor ahorro.

No obstante, se pueden implementar mejoras para obtener un mayor "speedup" como es el caso de implementar una versión paralela de rebalance de malla gruesa, con el cual se reduciría el número de iteraciones internas. También se optaría por implementar una versión hibrida con openMP para paralelizar la variable de energía con lo que finalmente se implementaría una versión tridimensional paralela.

AGRADECIMIENTOS

Los autores agradecen el apoyo financiero recibido del proyecto estratégico No. 212602 (AZTLAN Platform) del Fondo Sectorial de Sustentabilidad Energética CONACYT-SENER.

El primer autor agradece al Consejo Nacional de Ciencia y Tecnología (CONACYT) por la beca que está siendo otorgada para la realización de sus estudios de doctorado.

REFERENCIAS

1. "AZTLAN Platform", http://www.aztlanplatform.mx (2019).
2. Gómez Torres, A. M., Puente Espel, F., del Valle Gallegos, E., François, J. L., Martín-del-Campo, C. and Espinosa-Paredes, G. (2015). AZTLAN: Mexican Platform for Analysis and Design of Nuclear Reactors. In *Proceedings of ICAPP 2015*, Nice, France. May 03-06. Paper 15493.
3. Duran Gonzalez, Julian Arturo. *Implementación de la Cinética en el Código de Transporte Tridimensional AZTRAN*, tesis maestría, Instituto Politécnico Nacional, ESFM, cdmx (2017).
4. Duran Gonzalez Julian A., del Valle Gallegos Edmundo, Gómez Torres Armando M., "Implementación de la Cinética en el Código de Transporte AZTRAN", *Memorias XXVIII Congreso Anual de la Sociedad Nuclear Mexicana*, 18 al 21 de junio de 2017.
5. Evans Thomas, Stafford Alissa, Clarno Kevin, "Denovo—A New Three-Dimensional Parallel Discrete Ordinates Code in SCALE", *Nuclear technology*, **171**, p. 171-200 (2010).
6. R Alcouffe, R. S. Baker, J. A. Dahl, A. Turner, "PARTISN Code Abstract", *Physor 2000 International Topical Meeting, Advances in Reactor Physics and Mathematics and Computation into the Next Millennium*, Pittsburgh, USA, (2000).
7. T. Courau and G. Sjoden, "3D Neutron Transport and HPC: A PWR Full Core Calculation Using PENTRAN SN Code and IBM BLUEGENE/P Computers," *Progress in Nuclear Science and Technology*, **2**, 628–633 (2011).
8. K. Koch, R. Baker, R. Alcouffe, "Solution of the first-order form of the 3-D discrete ordinates equation on a massively parallel processor", *Transactions of the American Nuclear Society* 65 (1992) 198–199.
9. E. MASIELLO, B. MARTIN, and J.-M. DO, "Domain Decomposition and CMFD Acceleration Applied to Discrete-Ordinate Methods for the Solution of the Neutron Transport Equation in XYZ Geometries," *Proc. Int. Conf. Mathematics and Computational Methods (M&C2011)*, Rio de Janeiro, Brazil, May 8–12, 2011.
10. R. LENAIN et al., "Domain Decomposition Method for 2D and 3D Transport Calculations Using Hybrid MPI/OpenMP Parallelism," *Proc. Int. Conf. Mathematics and Computational Methods (M&C2015)*, Nashville,Tennessee, April 19–23, 2015.

11. Duran Gonzalez Julian A., del Valle Gallegos Edmundo, Gómez Torres Armando M., "Paralelización, en Grupos de Energía y Direcciones Angulares en el Código AZTRAN usando MPI", *Memorias XXIX Congreso Anual de la Sociedad Nuclear Mexicana*, 2 al 5 de julio de 2018.
12. Gropp, W., E. Lusk, and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message Passing Interface*, Cambridge, Massachusetts: MIT Press, (1994).
13. "Benchmark on Deterministic Transport Calculations Without Spatial Homogenization", www.oecd-nea.org/science/docs/2003/nsc-doc2003-16.pdf (2003).
14. R Alcouffe, R. S. Baker, J. A. Dahl, A. Turner, "PARTISN Results for the C5G7 MOX Benchmark Problems", *Physor 2002 International Conference on the New Frontiers of Nuclear Technology: Reactor Physics, Safety and High-Performance Computing*, Seoul, Korea, (2002).