Mejoras en la Versión Paralela del Código Termohidráulico de Subcanal COBRA-TF

Enrique Ramos[a], Agustín Abarca[b], Jose E. Roman[a], Rafael Miró[b]

[a]Dpto. de Sistemas Informáticos y Computación,
Universitat Politècnica de València,
Camí de Vera s/n, 46022 Valencia, España
ramos@dsic.upv.es, jroman@dsic.upv.es

[b] Inst. de Seguridad Industrial, Radiofísica y Medioambiental (ISIRYM)
Universitat Politècnica de València,
Camí de Vera s/n, 46022 Valencia, España

aabarca@isirym.upv.es, rmiro@iqn.upv.es

Resumen – *Realizar análisis en seguridad nuclear a nivel de varilla de combustible requiere las ejecución de códigos acoplados neutrónico-termohidráulicos que permitan la simulación de grandes dominios físicos en un tiempo razonable. Para ello resulta imprescindible el uso de numerosos procesadores (o núcleos) que colaboren en la obtención de la solución a un único problema utilizando la potencia computacional y de memoria disponible en un clúster.*

En anteriores trabajos se presentó la paralelización del código termohidráulico de subcanal COBRA-TF (CTF) mediante la tecnología MPI (Message Passing Interface) y utilizando la librería paralela de métodos numéricos PETSc para resolver el sistema lineal de ecuaciones del código. En el presente documento se expondrán las mejoras introducidas en código acoplado, centradas tanto en la parte de optimización de memoria como en paralelismo, además se ha combinado la tecnología MPI y la PVM (Parallel Virtual Machine) para posibilitar el uso del código acoplado CTF/PARCSV2.7.

1. INTRODUCCIÓN

Para realizar una paralelización efectiva de un código termohidráulico como CTF se debe paralelizar tanto la actualización de variables, como la generación del Jacobiano y la resolución del sistema lineal, gestionando la memoria de forma adecuada por los diferentes procesos.

En este trabajo se presentarán las mejoras introducidas en la versión paralela del código CTF, tanto en la gestión de la memoria como en la paralelización y comunicación de variables. Además, todas estas mejoras se validarán mediante la simulación de un caso acoplado del código CTF/PARCSV2.7 utilizando conjuntamente tecnología MPI y PVM para la paralelización y comunicación entre códigos respectivamente.

COBRA-TF (*Coolant Boiling in Rod Arrays Code – Two Fluids*), abreviado como CTF, es un código termohidráulico de subcanal (permite resolver problemas a nivel de varilla de

combustible nuclear) que utiliza dos campos y tres fases para modelar el flujo bifásico (agua+vapor) [1], [2]. Las tres fases son el vapor, el líquido continuo y el líquido presente a modo de gotas dentro del vapor en ciertos regímenes. Cada fase utiliza un conjunto de tres ecuaciones tridimensionales para la masa, momento y energía, con una excepción: se utiliza una ecuación de la energía para ambas fases líquidas, la de líquido y la de las gotas presentes en la fase vapor. La formulación de dos fluidos emplea un conjunto separado de ecuaciones de conservación y relaciones constitutivas para cada fase. Los efectos de una fase en otra se tienen en cuenta en los términos de interacción que aparecen en la correspondiente ecuación de gobierno. Las ecuaciones de conservación tienen la misma forma para cada fase; solamente se diferencian en las relaciones constitutivas y las propiedades físicas.

En este trabajo se presentan las mejoras añadidas a la paralelización del código CTF. Tras la integración con una librería paralela para resolución de sistemas de ecuaciones, concretamente PETSc [3], [4], [5] y la paralelización de la generación del Jacobiano, de modo que cada procesador se encarga de la generación de los coeficientes asociados a un subconjunto de celdas, el siguiente paso fue optimizar en la medida de lo posible el código para ahorrar memoria y tiempo de ejecución y sobre todo resolver el acoplamiento entre CTF y PARCSv2.7.

En las secciones 2 y 3 se repasan algunos detalles del código que son relevantes para la paralelización. Las secciones 4 y 5 presentan las mejoras realizadas al código. Por último la sección 6 nos ofrece resultados preliminares de ejecución.

2. RESOLUCIÓN NUMÉRICA

El primer paso de la resolución consiste en resolver las ecuaciones de momento de cada celda por eliminación Gaussiana, cuya forma matricial se muestra en la ecuación (1).

$$\begin{bmatrix} c_1 - 1 & d_1 & 0 \\ c_2 & d_2 - 1 & e_2 \\ 0 & d_3 & e_3 - 1 \end{bmatrix} \begin{Bmatrix} f_l \\ f_v \\ f_e \end{Bmatrix} = \begin{Bmatrix} -a_1 - b_1 \Delta P \\ -a_2 - b_2 \Delta P \\ -a_3 - b_3 \Delta P \end{Bmatrix} \tag{1}$$

El segundo paso nos permite calcular las velocidades, necesarias para la linealización de las ecuaciones de masa y energía. Mediante el método de Newton-Raphson [6] por bloques se obtiene la variación de las variables independientes que garantizan un error residual nulo, cuya ecuación podemos ver en la ecuación (2).

$$\begin{aligned} E_{\mathrm{CV}} = & \frac{\left[(\alpha_v \rho_v)_j^n - (\alpha_v \rho_v)_j\right] A_{c_j}}{\Delta t} + \sum_{\mathrm{KA}=1}^{\mathrm{NA}} \frac{\left[(\alpha_v \rho_v)^* \widetilde{U}_{v_j} A_{m_j}\right]_{\mathrm{KA}}}{\Delta x_j} \\ & - \sum_{\mathrm{KB}=1}^{\mathrm{NB}} \frac{\left[(\alpha_v \rho_v)^* \widetilde{U}_{v_{j-1}} A_{m_{j-1}}\right]_{\mathrm{KB}}}{\Delta x_j} - \sum_{L=1}^{\mathrm{NKK}} S_L \left[(\alpha_v \rho_v)^* \widetilde{V}_{v_L}\right]_j \\ & - \frac{\Gamma_j}{\Delta x_j} - \frac{S_{cv_j}}{\Delta x_j} \end{aligned} \tag{2}$$

$$\begin{bmatrix}
\frac{\partial E_{\mathrm{CG}}}{\partial \alpha_g} & \frac{\partial E_{\mathrm{CG}}}{\partial \alpha_v} & \frac{\partial E_{\mathrm{CG}}}{\partial \alpha_v h_v} & \frac{\partial E_{\mathrm{CG}}}{\partial (1-\alpha_v) h_l} & \frac{\partial E_{\mathrm{CG}}}{\partial \alpha_e} & \frac{\partial E_{\mathrm{CG}}}{\partial P_j} & \frac{\partial E_{\mathrm{CG}}}{\partial P_{i=1}} & \cdots & \frac{\partial E_{\mathrm{CG}}}{\partial P_{\mathrm{NCON}}} \\
\frac{\partial E_{\mathrm{CL}}}{\partial \alpha_g} & \frac{\partial E_{\mathrm{CL}}}{\partial \alpha_v} & \frac{\partial E_{\mathrm{CL}}}{\partial \alpha_v h_v} & \frac{\partial E_{\mathrm{CL}}}{\partial (1-\alpha_v) h_l} & \frac{\partial E_{\mathrm{CL}}}{\partial \alpha_e} & \frac{\partial E_{\mathrm{CL}}}{\partial P_j} & \frac{\partial E_{\mathrm{CL}}}{\partial P_{i=1}} & \cdots & \frac{\partial E_{\mathrm{CL}}}{\partial P_{\mathrm{NCON}}} \\
\frac{\partial E_{\mathrm{CV}}}{\partial \alpha_g} & \frac{\partial E_{\mathrm{CV}}}{\partial \alpha_v} & \frac{\partial E_{\mathrm{CV}}}{\partial \alpha_v h_v} & \frac{\partial E_{\mathrm{CV}}}{\partial (1-\alpha_v) h_l} & \frac{\partial E_{\mathrm{CV}}}{\partial \alpha_e} & \frac{\partial E_{\mathrm{CV}}}{\partial P_j} & \frac{\partial E_{\mathrm{CV}}}{\partial P_{i=1}} & \cdots & \frac{\partial E_{\mathrm{CV}}}{\partial P_{\mathrm{NCON}}} \\
\frac{\partial E_{\mathrm{CE}}}{\partial \alpha_g} & \frac{\partial E_{\mathrm{CE}}}{\partial \alpha_v} & \frac{\partial E_{\mathrm{CE}}}{\partial \alpha_v h_v} & \frac{\partial E_{\mathrm{CE}}}{\partial (1-\alpha_v) h_l} & \frac{\partial E_{\mathrm{CE}}}{\partial \alpha_e} & \frac{\partial E_{\mathrm{CE}}}{\partial P_j} & \frac{\partial E_{\mathrm{CE}}}{\partial P_{i=1}} & \cdots & \frac{\partial E_{\mathrm{CE}}}{\partial P_{\mathrm{NCON}}} \\
\frac{\partial E_{\mathrm{EL}}}{\partial \alpha_g} & \frac{\partial E_{\mathrm{EL}}}{\partial \alpha_v} & \frac{\partial E_{\mathrm{EL}}}{\partial \alpha_v h_v} & \frac{\partial E_{\mathrm{EL}}}{\partial (1-\alpha_v) h_l} & \frac{\partial E_{\mathrm{EL}}}{\partial \alpha_e} & \frac{\partial E_{\mathrm{EL}}}{\partial P_j} & \frac{\partial E_{\mathrm{EL}}}{\partial P_{i=1}} & \cdots & \frac{\partial E_{\mathrm{EL}}}{\partial P_{\mathrm{NCON}}} \\
\frac{\partial E_{\mathrm{EV}}}{\partial \alpha_g} & \frac{\partial E_{\mathrm{CG}}}{\partial \alpha_v} & \frac{\partial E_{\mathrm{EV}}}{\partial \alpha_v h_v} & \frac{\partial E_{\mathrm{EV}}}{\partial (1-\alpha_v) h_l} & \frac{\partial E_{\mathrm{EV}}}{\partial \alpha_e} & \frac{\partial E_{\mathrm{EV}}}{\partial P_j} & \frac{\partial E_{\mathrm{EV}}}{\partial P_{i=1}} & \cdots & \frac{\partial E_{\mathrm{EV}}}{\partial P_{\mathrm{NCON}}}
\end{bmatrix}$$

$$\cdot \begin{Bmatrix}
\delta\alpha_g \\ \delta\alpha_v \\ \delta\alpha_v h_v \\ \delta(1-\alpha_v) h_l \\ \delta\alpha_e \\ \delta P_j \\ \delta P_{i=1} \\ \vdots \\ \delta P_{i=\mathrm{NCON}}
\end{Bmatrix} = - \begin{Bmatrix}
E_{\mathrm{CG}} \\ E_{\mathrm{CL}} \\ E_{\mathrm{CV}} \\ E_{\mathrm{CE}} \\ E_{\mathrm{EL}} \\ E_{\mathrm{EV}}
\end{Bmatrix} \qquad (3)$$

Aplicando Newton-Raphson podemos obtener las ecuaciones matriciales para cada celda que se muestran en la ecuación (3) en la que podemos ver el Jacobiano del sistema de ecuaciones evaluado para las variables independientes, y compuesto por las derivadas analíticas de cada ecuación con respecto a la variación lineal de las variables independientes. También podemos observar el vector solución que contiene las variaciones lineales y el vector de error.

Una vez resuelta la ecuación se reduce para obtener la solución para las variables independientes, derivando una ecuación para cada celda, lo que obliga a resolver un sistema de ecuaciones formado por todas las celdas. Si se alcanza la convergencia se pasa al paso de tiempo siguiente, de lo contrario se disminuye el paso de tiempo de la simulación y se repite el proceso.

En resumen, cada iteración externa se compone principalmente de dos partes: el cálculo de los coeficientes y la resolución de un sistema de ecuaciones lineales. La matriz de coeficientes de este sistema de ecuaciones es dispersa (el porcentaje de elementos no nulos es muy bajo). Esto se debe al método de discretización utilizado, concretamente se trata del método de diferencias finitas. Es importante aprovechar esta característica para resolver el sistema eficientemente, utilizando métodos iterativos. En este trabajo, se ha abordado la paralelización de la resolución del sistema y del cálculo de los coeficientes mediante el reparto de las celdas del modelo entre varios procesos MPI.

3. PETSC

PETSc (*Portable Extensible Toolkit for Scientific Computation*), es un software numérico orientado a objetos para la resolución de ecuaciones en derivadas parciales, paralelizado mediante paso de mensajes. Está siendo utilizado en muchas aplicaciones científicas en todo el mundo. En PETSc todo el código se construye alrededor de una serie de estructuras de datos y algoritmos que han sido encapsuladas mediante técnicas de orientación a objetos. Su ventaja consiste en trabajar directamente con estos objetos abstrayéndose de la estructura de datos interna, lo que le da una gran potencia al permitir programar los métodos numéricos y aplicaciones habituales sin tener que estar pendiente de multitud de detalles de

implementación relativos a las estructuras de datos o la paralelización. En la figura 1 se pueden ver los principales componentes de PETSc.

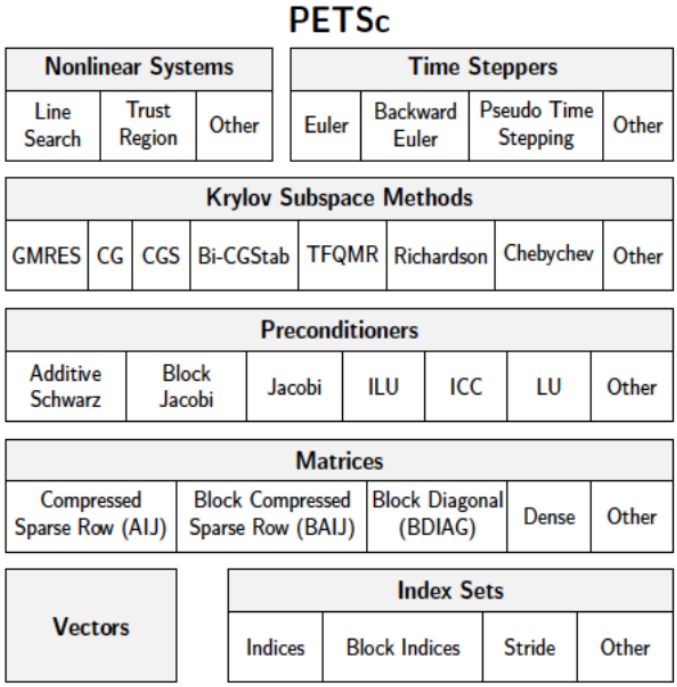

Figura 1. Principales componentes de PETSc.

4. OPTIMIZACIÓN DE LA MEMORIA Y EL TIEMPO DE EJECUCIÓN

Ante la envergadura de las simulaciones, que implican unas cantidades de memoria y de tiempo de ejecución realmente importantes, una vez concluida la paralelización y comprobada su corrección, se procedió a reducir al máximo ambos factores.

El primer paso fue tratar de reducir en la medida de lo posible la memoria necesaria para poder ejecutar el código. Este factor es fundamental en los casos de simulación muy grandes ya que requieren grandes cantidades de memoria para almacenar los datos. Este aspecto es mucho más importante si tenemos en cuenta que en las ejecuciones paralelas cada proceso necesita su espacio de memoria. En este sentido se trabajó en los siguientes aspectos:

- Análisis de los *inputs* para eliminar datos no necesarios y reducir al máximo algunos datos que estaban dimensionados de forma exagerada innecesariamente. Con esto se consiguió que varias variables que estaban dimensionadas a grandes tamaños, se adaptasen a las necesidades reales de cada ejecución.
- Análisis de las variables locales para eliminar algunas no necesarias en todas las ejecuciones y dimensionar de forma dinámica al input en la medida de lo posible.

Con estas dos actuaciones se consiguió reducir drásticamente las necesidades de memoria de cada proceso en las ejecuciones paralelas.

El segundo aspecto a disminuir era el tiempo de ejecución de cada proceso. Para ello se realizó una lectura y optimización exhaustiva de todas las subrutinas, especialmente centrada en los siguientes aspectos:

- Eliminación de bucles innecesarios y condicionamiento a los valores de los *inputs* de otros bucles, de forma que sólo se ejecuten si es necesario por el tipo de simulación.

- Reordenamiento de bucles para tratar de eliminar anidamientos en la medida de lo posible.
- Revisión de los recorridos de los bucles, muchos de ellos mal dimensionados a variables que podían ser demasiado grandes en algunos casos.

Estas actuaciones en algunos casos permitieron reducir a la mitad el tiempo de ejecución de todas las subrutinas involucradas.

5. ACOPLE DE CTF Y PARCSV2.7

Una vez reducidas las necesidades temporales y espaciales al máximo, el otro aspecto fundamental a resolver era el acoplamiento entre CTF y PARCSv2.7. Para ello hubo que estudiar detenidamente el acople secuencial realizado con PVM y adaptarlo al código paralelo mediante MPI.

La tarea consistió en combinar ambos ya que no se quería sustituir toda la política realizada con PVM y que estaba en constante modificación. Para ello mediante MPI se integró de forma que todos los procesos paralelos se acoplasen con el código PARCSv2.7. De esta forma se ha conseguido la comunicación entre ambos códigos de forma eficaz y en paralelo con la ganancia temporal que ello implica en las ejecuciones. En la figura 2 puede observarse de manera esquemática el acople del código paralelo.

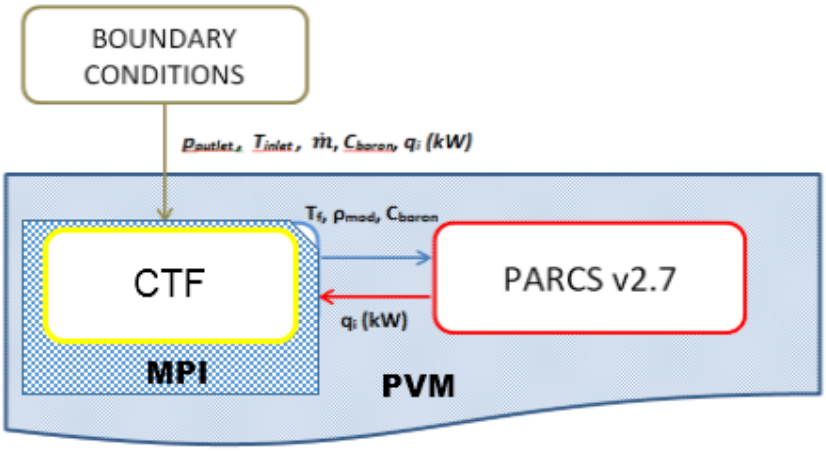

Figura 2. Esquema de acople del código paralelo CTF con PARCSv2.7.

Para facilitar la labor, el código permite realizar copia a disco del estado de las variables de cada proceso para posteriormente cargarlas y poder continuar, todo ello en paralelo. El procedimiento seguido ha sido el siguiente:

- Realizar una simulación no acoplada, ejecutando sólo CTF hasta conseguir una estabilización de los resultados.
- Guardar en disco el estado de las variables en ese momento.
- Cargar de disco el estado guardado y realizar una simulación acoplada entre CTF y PARCSv2.7 con datos ya estabilizados y un nuevo *input* que tenga en cuenta en acople.

6. RESULTADOS

A título demostrativo, en este apartado se presentarán los resultados obtenidos de la simulación del estado estacionario del núcleo de un reactor PWR de tres lazos, cuyo modelo de CTF se ha realizado varilla a varilla, incluyendo en los resultados la aceleración obtenida con respecto al caso secuencial y la eficiencia del código paralelo.

En las figuras 3 y 4 se presentan respectivamente la entalpía y la distribución de caudales a la salida del núcleo a nivel de subcanal. Como puede apreciarse en las figuras 3 y 7, tal como es de esperar, se obtiene una distribución simétrica en los cuatro cuadrantes del núcleo, distinguiendo claramente la posición de los tubos guía en el interior de los elementos combustibles.

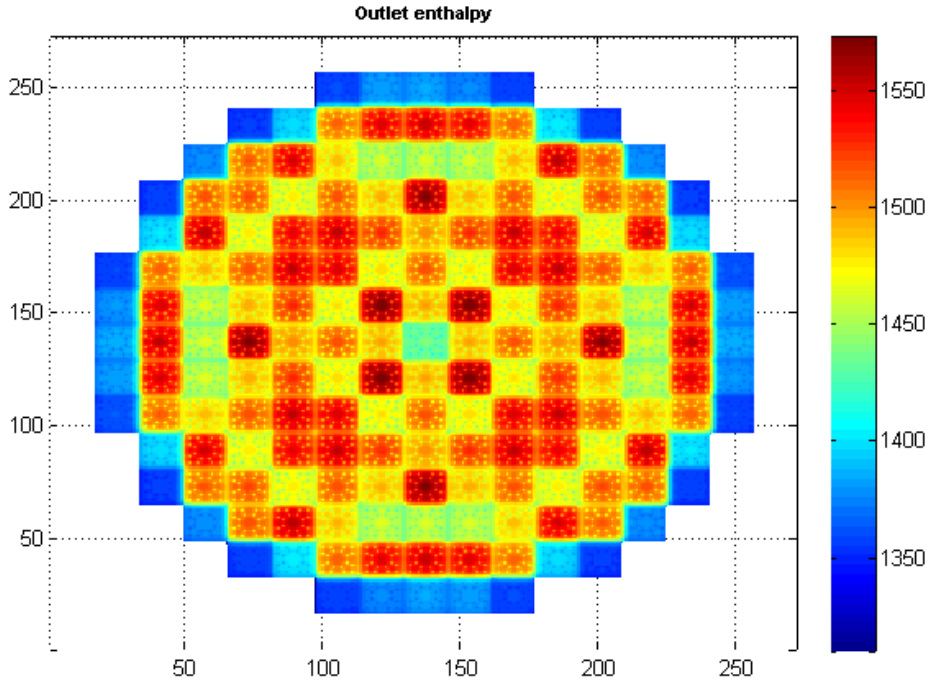

Figura 3. Mapa radial de la entalpía a la salida del núcleo.

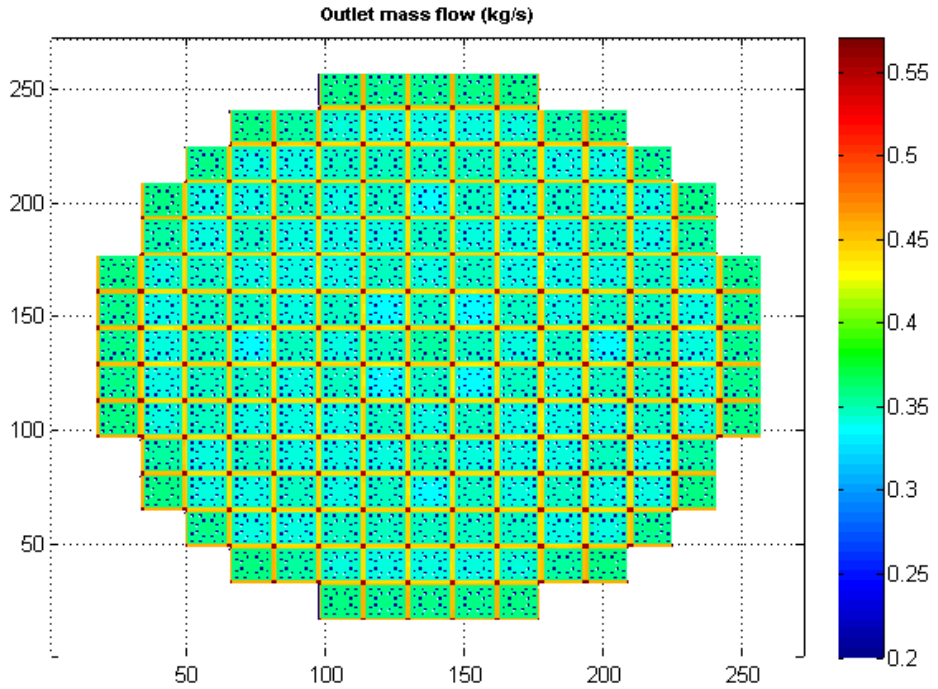

Figura 4. Mapa radial de caudales a la salida del núcleo.

En la tabla 1 se presentan los resultados obtenidos en cuanto a aceleración del cálculo y eficiencia de la paralelización de CTF. Como puede observarse se pasa de unos 133s en el caso secuencial a unos 29s utilizando 7-8 procesos MPI, siendo la eficiencia del paralelismo adecuada (superior al 75%) hasta 5 procesos, donde se pierde la escalabilidad del cálculo.

Tabla 1. Esquema de acople del código paralelo CTF con PARCSv2.7.

Número de procesos MPI	Tiempo total de simulación (s)	Aceleración	Eficiencia (ts – tp) / np
1	133.778	----	----
2	73.865	1.811	0.906
3	52.522	2.547	0.849
4	43.055	3.107	0.777
5	35.066	3.815	0.763
6	32.133	4.163	0.694
7	29.272	4.570	0.653
8	29.431	4.546	0.568

Por último, en las figuras 5 y 6 se presenta la evolución de la aceleración y la eficiencia conseguida en la paralelización de CTF para el caso de estudio. Como puede observarse existe escalabilidad en el cálculo hasta 5 procesos, donde la pendiente de la aceleración disminuye y la eficiencia de código cae en mayor proporción.

Para explicar la pérdida de escalabilidad cabe mencionar que CTF se ha paralelizado en subdominios axiales, y que el modelo utilizado para este ejemplo cuenta únicamente con 32 nodos axiales. Por tanto si utilizamos 8 procesos se está asignando únicamente 4 nodos a cada proceso y debido a la necesidad de comunicación de variables entre distintos procesos, se ve limitada la eficiencia del código.

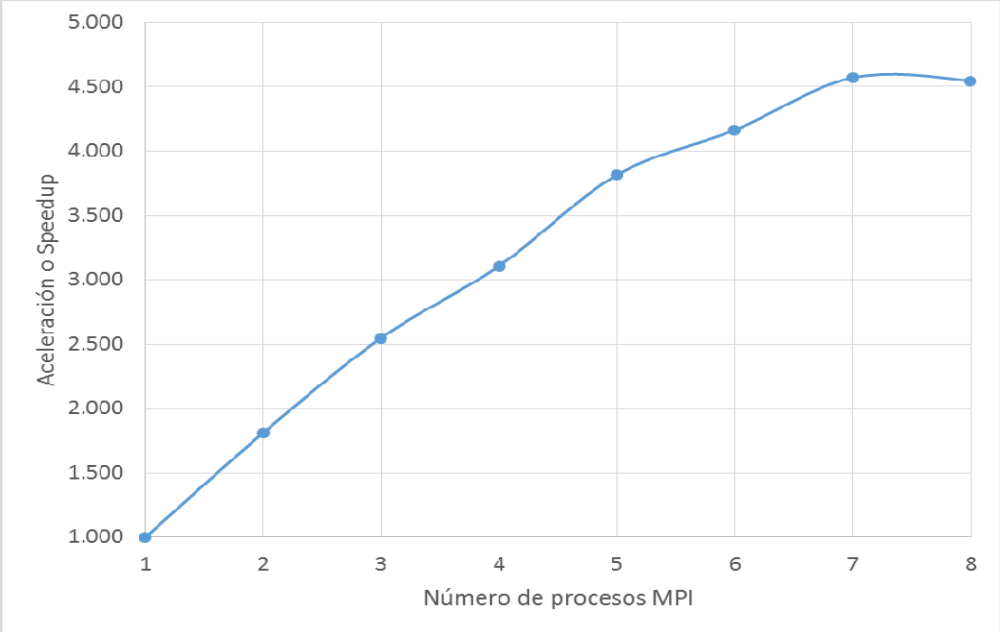

Figura 5. Mapa radial de caudales a la salida del núcleo.

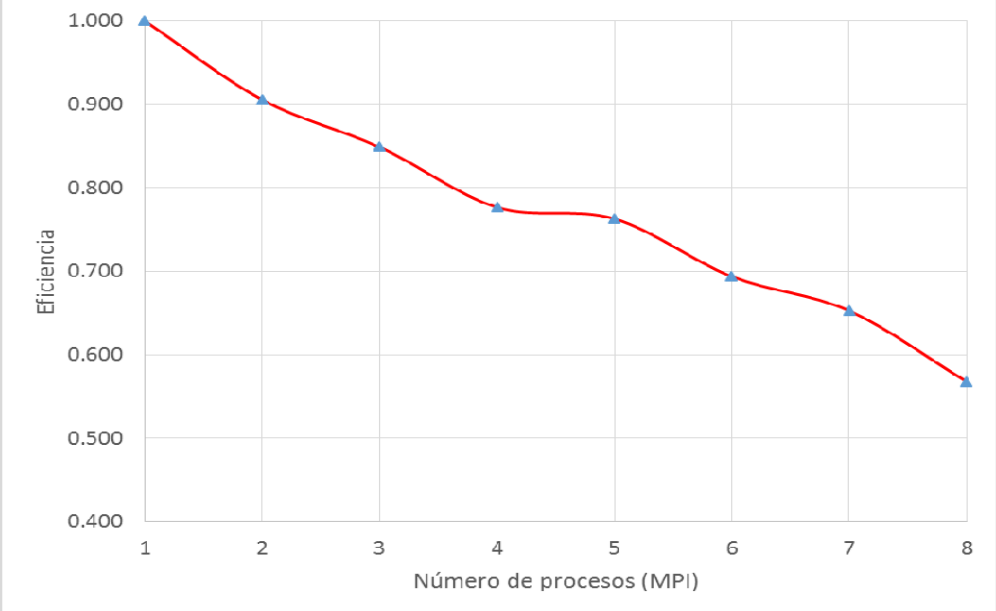

Figura 6. Mapa radial de caudales a la salida del núcleo.

7. CONCLUSIONES

Este proyecto ha sido desarrollado con el objetivo de aplicar las más modernas técnicas de ingeniería de informática, como son los paradigmas de la computación en paralelo, a los códigos de simulación termohidráulica existentes, como el código de subcanal CTF. Centrándose el presente trabajo en las mejoras realizadas en el código paralelo durante el último ejercicio, incluyendo la utilización del código paralelo acoplado con PARCSv2.7.

Todas estas mejorar en la gestión de la memoria y el funcionamiento en paralelo ha sido verificada mediante la simulación del estado estacionario de un núcleo PWR varilla a varilla

en tiempos razonables, obteniendo unos resultados esperables en este tipo de plantas nucleares y mostrando las virtudes del código paralelo CTF.

AGRADECIMIENTOS

Este trabajo ha sido parcialmente esponsorizado por la *Universitat Politècnica de València* bajo los proyectos COBRA_PAR PAID-05-11-2810 y OpenNUC PAID-05-12 y por el *Ministerio de Economía y Competitividad* a través del proyecto NUC-MULTPHYS ENE-2012-34585 financiado por los fondos FEDER de la Unión Europea.

REFERENCIAS

1) Diana Cuervo, Improving the Computation Efficiency of COBRA-TF for LWR Safety Analysis of Large Problems, PHYSOR, Chicago, USA, 2004.
2) Diana Cuervo, *Implementation and performance of Krylov Methods for the solution of Two-Fluid Hydrodinamics Equations in the COBRA-TF Code*, M&C, Avignon, France, 2005.
3) Satish Balay, Jed Brown, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, Hong Zhang, PETSc Web page, "http://www.mcs.anl.gov/petsc", 2011.
4) Satish Balay, Jed Brown, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, Hong Zhang, PETSc User's Manual, ANL-95/11 - Revision 3.3, Argonne National Laboratory, 2012.
5) Satish Balay, William D. Gropp, Dinesh Kaushik, Lois Curfman McInnes, Barry F. Smith, Efficient Management of Parallelism in Object Oriented Numerical Software Libraries, E. Arge and A. M. Bruaset and H. P. Langtangen, 1997.