Análisis Comparativo del Cálculo Numérico Serie/Paralelo de la Termohidráulica de Canales con Ebullición

Miguel Ceceñas Falcón
Instituto Nacional de Electricidad y Energías Limpias
Reforma 113 Col. Palmira. 62490 Cuernavaca, Mor.
mcf@iie.org.mx

Resumen

Se emplea un modelo de canales en paralelo con ebullición y cinética neutrónica puntual para comparar la implementación de su programación en lenguaje C mediante un esquema convencional y mediante un esquema de programación en paralelo. En ambos casos las subrutinas escritas en C son prácticamente las mismas, pero varían en la manera de controlar la ejecución de las tareas que calculan los diferentes canales. Para la solución en paralelo se emplea *Parallel Virtual Machine* (PVM), que permite el paso de mensajes entre tareas para controlar convergencia y transferir las variables de interés entre las tareas que se ejecutan simultáneamente en una plataforma equipada con microprocesador multi-núcleo. Para algunos problemas definidos como caso de estudio, tal como el presentado en el presente trabajo, una computadora con dos núcleos puede reducir el tiempo de cómputo al 54-56% del tiempo necesario por el mismo programa en su versión secuencial convencional. Similarmente, un procesador con cuatro núcleos puede reducir el tiempo hasta un 22-33% del tiempo de ejecución de la versión serie convencional. Estos resultados de reducir sustancialmente el tiempo de cómputo son muy motivantes para todas aquellas aplicaciones que puedan prepararse para ser paralelizadas y cuyo tiempo de ejecución sea un factor importante.

.

1. INTRODUCCIÓN

El desarrollo actual de los microprocesadores de las computadoras se basa cada vez menos en el incremento en su velocidad de reloj, incrementando en su lugar la densidad de los transistores integrados para crear procesadores multi-núcleo. El cálculo con multinúcleos impacta todo el espectro de aplicaciones, desde problemas científicos complejos hasta aplicaciones comerciales, por lo que la programación en paralelo presenta a la vez un amplio espectro de aplicaciones y una nueva opción para la mayoría de los desarrolladores.

La idea del paralelismo no es nueva, pero su desarrollo en general estaba sujeto a la dificultad de encontrar hardware adecuado con presupuestos modestos. Como soluciones de costo moderado, en los años 90s se construyeron redes heterogéneas de computadoras comunicadas mediante red ethernet y controladas por software tal como Parallel Virtual Machine (PVM) o implementaciones

del protocolo Message Passing Interface (MPI), logrando éxitos notorios al aplicar programación en paralelo a problemas que eran exclusivos de supercomputadoras de costo sustancialmente mayor. Como desventaja de estos sistemas, conocidos como *Beowulf clusters*, se tiene que las aplicaciones que requieren continua transferencia de información entre procesos presentaron problemas con la capacidad de la red, lo cual impacta la velocidad de cómputo. Actualmente, con los núcleos disponibles en un microprocesador común, no es necesario el uso de una red para lograr un sistema con cuatro u ocho CPUs o núcleos trabajando simultáneamente en un solo problema.

Al enfocarse en una aplicación particular, el grado de paralelización depende mucho del problema a resolver. Existen problemas que de manera natural se pueden paralelizar, tal como el cálculo termohidráulico en los ensambles de combustible en el núcleo de un reactor de agua en ebullición BWR. Otro ejemplo es el cálculo de estabilidad mediante el procesamiento de las mediciones de los detectores de flujo neutrónico en el núcleo de un reactor. La instrumentación de flujo neutrónico en un reactor tipo BWR se basa en un conjunto de detectores distribuidos en el volumen del núcleo, y el cálculo de estabilidad se puede realizar para cada uno de los detectores de manera independiente y en paralelo. En general, cualquier cálculo numéricamente demandante de CPU que sea repetitivo puede ser realizado mediante un grupo de tareas independientes, las cuales pueden ser ejecutadas de manera simultánea, esto es, en paralelo, por los diferentes núcleos con que cuenta la plataforma de cómputo usada.

2. HERRAMIENTAS PARA PARALELISMO EN UNIX

Las primeras computadoras contaban con una sola unidad central de procesamiento (CPU), el cual ejecutaba secuencialmente, una tras otra, las diversas tareas que le eran requeridas. Interconectando varias de estas computadoras, se lograba un sistema distribuido, en donde se comparten recursos, de manera que una computadora puede acceder a recursos conectados a otra computadora. La computación que se lleva a cabo en un sistema distribuido se conoce como computación distribuida, y si a las computadoras de un sistema distribuido se les coordina con el software adecuado para trabajar en la solución de un mismo problema numérico, se llega a la computación en paralelo.

Existen diversas herramientas para realizar computación en paralelo en Unix, entre ellas se encuentran PVM y MPI [1,2]. Ambas herramientas trabajan el esquema de memoria distribuida, y toda su información se transmite mediante mensajes entre las diferentes tareas que se ejecutan de manera simultánea.

2.1. La especificación MPI

La Interfaz para Transferencia de Mensajes (MPI, por sus siglas en inglés) consiste en la especificación de una librería orientada al cálculo en paralelo y/o distribuido. Existen varias implementaciones de esta especificación, algunas de libre distribución [2]. Al igual que PVM, es un esquema de procesamiento en paralelo con memoria distribuida, cuyas aplicaciones se pueden desarrollar en C++ o Fortran.

2.2. La Máquina Virtual en Paralelo PVM

La Máquina Virtual en Paralelo (Parallel Virtual Machine o PVM, por sus siglas en inglés) consiste en una computadora virtual constituida por un grupo de computadoras interconectadas mediante red ethernet. Las computadoras pueden ser de diferentes arquitecturas, esto es, pueden ser computadoras personales con sistema operativo linux o estaciones de trabajo con sistema operativo UNIX de diferentes proveedores. El software PVM permite que este tipo de redes heterogéneas puedan ser usadas como una sola computadora en paralelo que potencia las capacidades de cada computadora individual.

El desarrollo de PVM inició en 1989 en Oak Ridge National Laboratory, con la participación de varias universidades. El producto obtenido, un sistema que permite iniciar tareas en diferentes computadoras y comunicar a las tareas entre sí, es de interés a la comunidad científica y está disponible como software de libre distribución.

PVM básicamente consiste en un proceso que se ejecuta en segundo plano, inicia y configura a la máquina virtual, y se apoya en una serie de librerías con la cuales se debe ligar el programa del usuario y con la cual controla a la máquina virtual. PVM está desarrollado para ligarse a programas del usuario escritos en C o Fortran.

3. APLICACIÓN A CANALES PARALELOS

Para realizar un análisis comparativo serie/paralelo, se parte de un modelo ya desarrollado con anterioridad [3], que representa al núcleo de un reactor BWR mediante un conjunto de 36 canales termohidráulicos en paralelo acoplados a cinética neutrónica modal. Por simplicidad, en el presente trabajo se usa cinética neutrónica puntual. Los canales se definen mediante segmentos de anillos de ensambles en el núcleo, de manera que los primeros canales (1 a 8) representan el centro del núcleo y los últimos (29 a 36) la periferia. A este modelo se le hacen las adecuaciones necesarias para ejecutarlo desde una implementación tradicional en serie, y también mediante la implementación de un esquema de computación en paralelo. La geometría se obtiene de un problema ya conocido en el medio, como lo es el Benchmark de Ringhals [4]. El núcleo de 648 ensambles se colapsa a 36 grupos de ensambles, cada grupo con un perfil axial de potencia equivalente, un flujo y un área de flujo equivalente. Para el objetivo del trabajo, que es evaluar el tiempo de cómputo numérico, se selecciona el caso simple de una perturbación mediante un pulso de reactividad al estado estacionario que corresponde al ciclo 14, caso 9 del Benchmark.

El modelo resuelve las ecuaciones de conservación para cada canal termohidráulico, y considera tres regiones, iniciando con una región de líquido subenfriado, que al avanzar el fluido axialmente entra a una zona de ebullición subenfriada, y finamente pasa a una zona de ebullición de bulto [3]. El modelo no se detallará en este trabajo por haber sido ya tratado en trabajos previos, pero adicionalmente a las adecuaciones para su comparación serie/paralelo, contiene modificaciones para facilitar su posible futuro acoplamiento a un cálculo detallado de la neutrónica desarrollado por terceros.

Para la implementación en paralelo, se selecciona el esquema maestro-esclavo [5], en el cual se inician 36 tareas, designadas como esclavos, que en coordinación con un programa de control, el maestro, reciben en cada paso de integración la potencia calculada por la neutrónica para cada

canal y generan los perfiles de temperatura y de fracciones de vacío necesarios para actualizar la retroalimentación de reactividad para la neutrónica puntual. La tarea maestra gestiona la comunicación entre las otras tareas y tiene a su cargo el control de la ejecución de manera sincronizada.

Para obtener el estado estacionario inicial con el esquema maestro/esclavos, el flujo en cada canal se inicializa distribuyendo el flujo total conocido entre los grupos ponderados por el número de ensambles por canal. Se itera calculando la caída de presión en cada canal y actualizando el flujo hasta obtener una caída de presión igual para todos los canales.

Las subrutinas de la termohidráulica son las mismas para las implementaciones serie y paralelo, pero el esquema de su uso cambia de manera importante. El paralelismo se emplea como la ejecución de una tarea para cada canal, y es PVM quien se encarga de distribuirla a los diferentes núcleos disponibles en la máquina virtual. La versión serie ejecuta secuencialmente los 36 canales y no aprovecha la disponibilidad de más de un núcleo en el sistema.

3.1. Implementación tradicional en serie

Para calcular el flujo por canal, el flujo se inicializa con el promedio aritmético ponderado por el número de ensambles en cada canal. Se calcula la caída de presión en cada canal, así como el promedio de todos los canales, y al flujo de cada canal se le suma una corrección iterativamente de manera que se reduzca la diferencia entre la caída de cada canal y la caída promedio de todos los cansales. Las Figuras 1 y 2 muestran la convergencia del flujo para igualar la caída de presión en todos los canales. En general, se observa que el flujo inicial tiende a incrementarse o decrementarse de manera suave hasta llegar después de un número de iteraciones a un valor que representa el estado estacionario con la misma caída de presión en todos los canales.

En las Figuras 1 y 2 se evita una gráfica demasiado cargada de información al presentar los 36 canales en 4 grupos. Es interesante observar que los ensambles correspondientes al anillo central-exterior (canales 19 a 27) conducen un flujo mayor que el promedio.

La Figura 3 muestra el flujo calculado en estado estacionario en cada canal, mediante una línea azul. El caso 9 del ciclo 14 del Benchmark [4] contiene información del flujo para cada uno de los ensambles del núcleo, por lo que es posible sumar el flujo de todos los ensambles que conforman cada uno de los 36 canales. La línea en rojo de la Figura 3 muestra el flujo para cada canal obtenido de los datos del Benchmark, y se observa una buena concordancia de los datos de flujo con los calculados por los canales.

Una vez obtenido el estado estacionario inicial, se perturba el sistema mediante un pulso de reactividad y se analiza su respuesta dinámica. La Figura 4 muestra la respuesta dinámica de la potencia normalizada obtenida mediante las ecuaciones de cinética puntual acopladas a los canales. Se obtiene una respuesta oscilatoria característica de un sistema con alta razón de decaimiento.

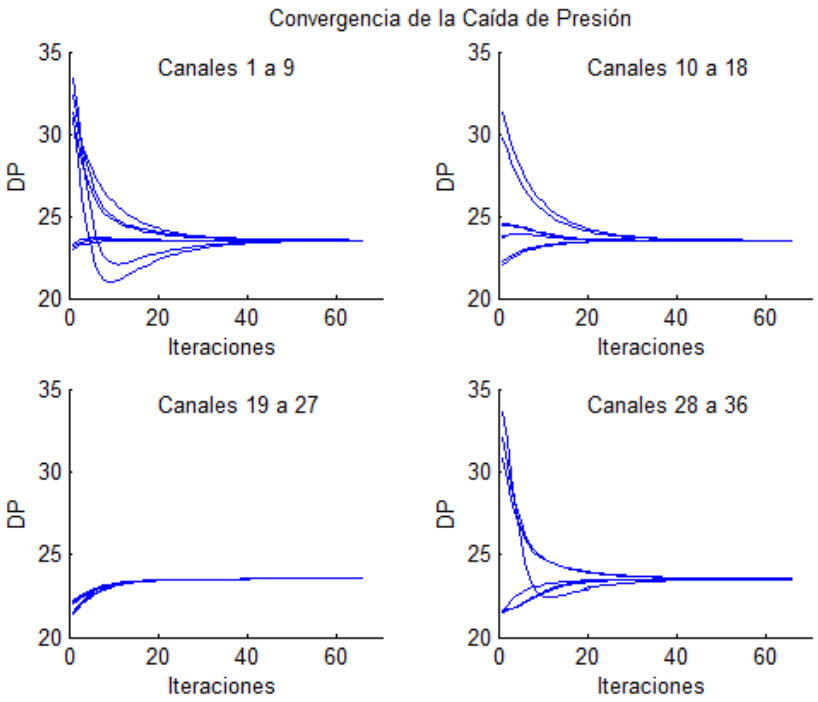

Figura 1. Convergencia de la caída de presión mediante iteraciones de flujo

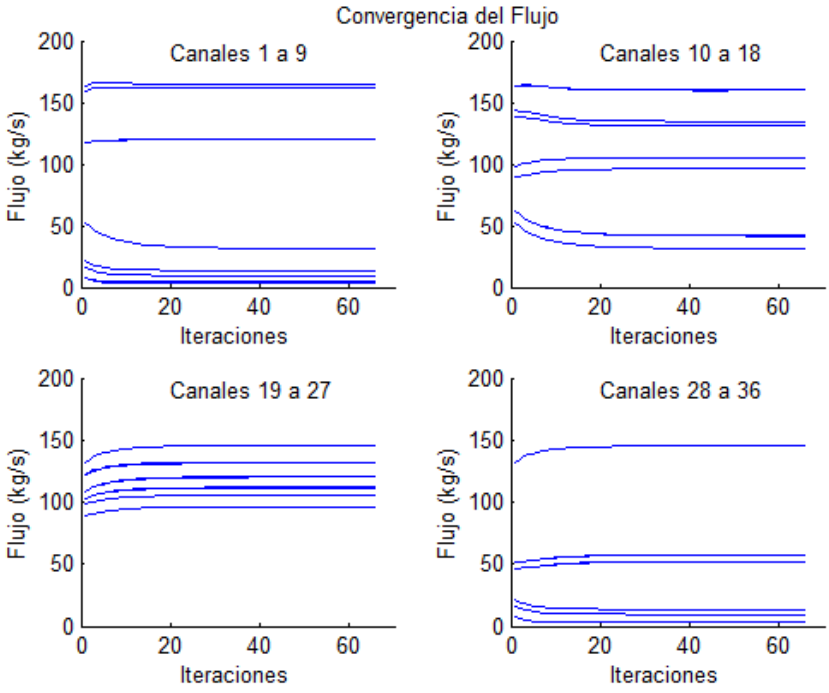

Figura 2. Convergencia del flujo en los canales termohidráulicos

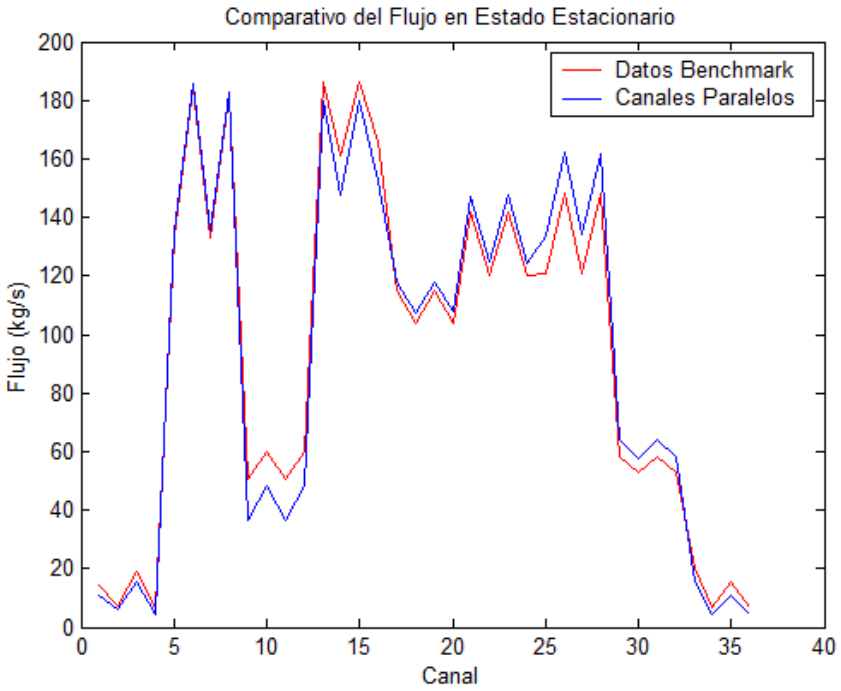

Figura 3. Distribución de flujo en los canales

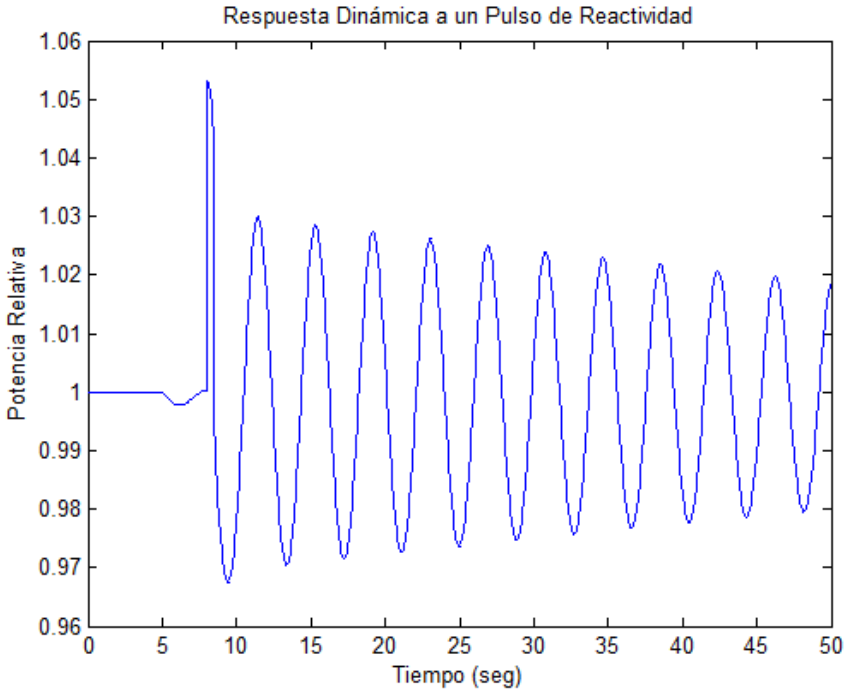

Figura 4. Respuesta dinámica a un pulso de reactividad

3.2. Implementación en paralelo mediante PVM

En cómputo distribuido, el esquema maestro/esclavo se usa ampliamente para implementar soluciones numéricas [1,5], por lo que este esquema se emplea para la implementación de los canales. El programa maestro se encarga de iniciar tareas o procesos esclavos, así como de proveerlos de sus datos y colectar sus resultados. Para su implementación, el programa original serie se divide en dos programas independientes que se comunican entre sí. El canal maestro lleva el control y calcula la neutrónica, mientras que el programa esclavo se ejecuta en 36 instancias para calcular la parte termohidráulica en cada canal. El maestro envía la potencia generada a cada canal, y cada esclavo actualiza temperaturas y fracciones de vacíos al maestro para calcular la retroalimentación y actualizar la neutrónica. Esta última se transfiere de nuevo a los esclavos cerrando así el ciclo.

3.3. Comparación de resultados

Para realizar las pruebas de ejecución, se consideró un grupo de 4 computadoras personales, listadas en la Tabla I, dos de ellas equipadas con procesadores con dos núcleos, y las otras dos con procesadores de 4 núcleos. Todas cuentan con sistema operativo Linux Fedora, exceptuando el equipo 4, con Linux Mint. El problema de los canales se ejecutó en repetidas ocasiones para comprobar consistencia en los tiempos de ejecución. Cada caso se ejecutó arbitrariamente en cinco ocasiones. Las Tablas II a V muestran resultados para los equipos listados en la Tabla I. Los tiempos de ejecución se expresan en milisegundos, y la columna "Par/Ser" muestra la relación del tiempo paralelo/serial.

En general, se observa consistencia en las pruebas realizadas, y es notoria la reducción en el tiempo de cómputo al disponer de mayor número de núcleos. De los equipos probados, el microprocesador de tecnología más reciente y con mayor número de núcleos y subprocesos obtuvo los mejores tiempos, aunque su velocidad de reloj no es la mayor.

Tabla I. Computadoras disponibles para evaluar las soluciones seriales y paralelas

Equipo	Procesador	Núcleos/Subp	RAM	Sistema Operativo
1	i5-2500 @ 3.3 GHz	4/4	4 GB	Linux Fedora 22 , 64 bit
2	i7-4720HQ @ 2.6 GHz	4/8	16 GB	Linux Fedora 25 , 64 bit
3	E8400 @ 3GHz	2/2	4 GB	Linux Fedora 22 , 64 bit
4	E5300 @ 2.6 GHz	2/2	4 GB	Linux Mint 17.2 , 64 bit

Tabla II. Resultados con el Equipo 1 con 4 núcleos

Prueba	Equipo 1: Sistema con 4 núcleos		
	Serie (ms)	Paralelo (ms)	Par/Ser
1	104.78	34.55	0.3297
2	104.91	37.06	0.3532
3	105.01	34.27	0.3264
4	105.23	34.27	0.3257
5	106.31	34.47	0.3243
Promedio	105.25	34.92	0.3318

Tabla III. Resultados con el Equipo 2 con 4 núcleos y 8 subprocesos

Prueba	Equipo 2: Sistema con 4 núcleos y 8 subp.		
	Serie (ms)	Paralelo (ms)	Par/Ser
1	93.26	21.26	0.2279
2	92.57	20.64	0.2230
3	92.88	20.64	0.2223
4	92.86	20.68	0.2227
5	92.82	20.57	0.2217
Promedio	92.88	20.76	0.2235

Tabla IV. Resultados con el Equipo 3 con 2 núcleos

Prueba	Equipo 3: Sistema con 2 núcleos		
	Serie (ms)	Paralelo (ms)	Par/Ser
1	157.77	85.02	0.5388
2	157.45	85.21	0.5412
3	157.86	85.61	0.5423
4	157.59	85.60	0.5432
5	157.32	85.34	0.5424
Promedio	157.60	85.36	0.5416

Tabla V. Resultados con el Equipo 4 con 2 núcleos

Prueba	Equipo 4: Sistema con 2 núcleos		
	Serie (ms)	Paralelo (ms)	Par/Ser
1	188.55	106.37	0.5641
2	190.32	104.99	0.5516
3	188.31	105.24	0.5588
4	188.48	105.01	0.5572
5	188.72	105.04	0.5566
Promedio	188.87	105.33	0.5577

La utilería de Linux "GNOME System Monitor" permite visualizar el uso de los recursos del sistema (procesador, memoria y red) mediante una interfaz gráfica. Para el caso de la implementación en paralelo, la Figura 5 muestra el uso de CPU reportado por esta utilería en un sistema con 4 núcleos, donde se observa que inicialmente éstos se encuentran inactivos, para luego aplicarse todos ellos en un pico de actividad relacionado a la convergencia del flujo para igualar la presión en los canales. Al terminar de iterar el flujo, el programa despliega en pantalla resultados y espera que el usuario oprima una tecla para calcular la respuesta a un pulso de reactividad. Esto se observa como un período sin uso de CPU de aproximadamente 3 segundos, seguido por un período de 38 segundos aproximadamente donde todos los núcleos participan en el cálculo del transitorio. Al finalizar el cálculo los núcleos vuelven a quedar inactivos.

Para el caso de la implementación serial, la Figura 6 ilustra la manera en la que el sistema operativo asigna un solo núcleo al cálculo de los canales. Se observa que después de un tiempo, el núcleo que se encuentra trabajando es relevado por uno de los núcleos que se encuentran libres. En la programación convencional o serial, cuando un núcleo trabaja al 100%, los núcleos restantes se encuentran libres.

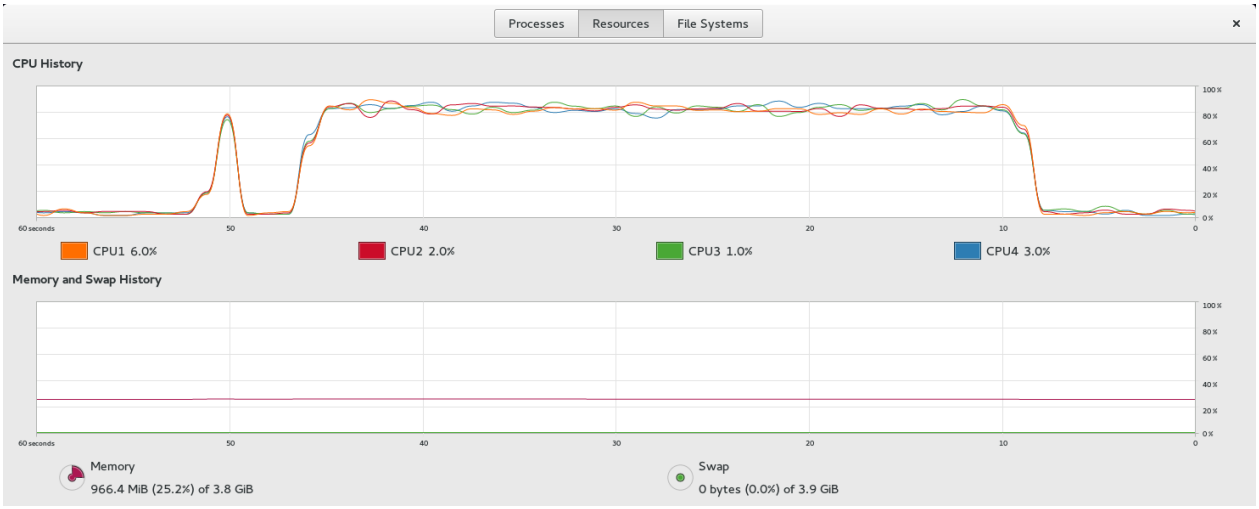

Figura 5. Uso de CPU durante la ejecución en paralelo de los canales termohidráulicos

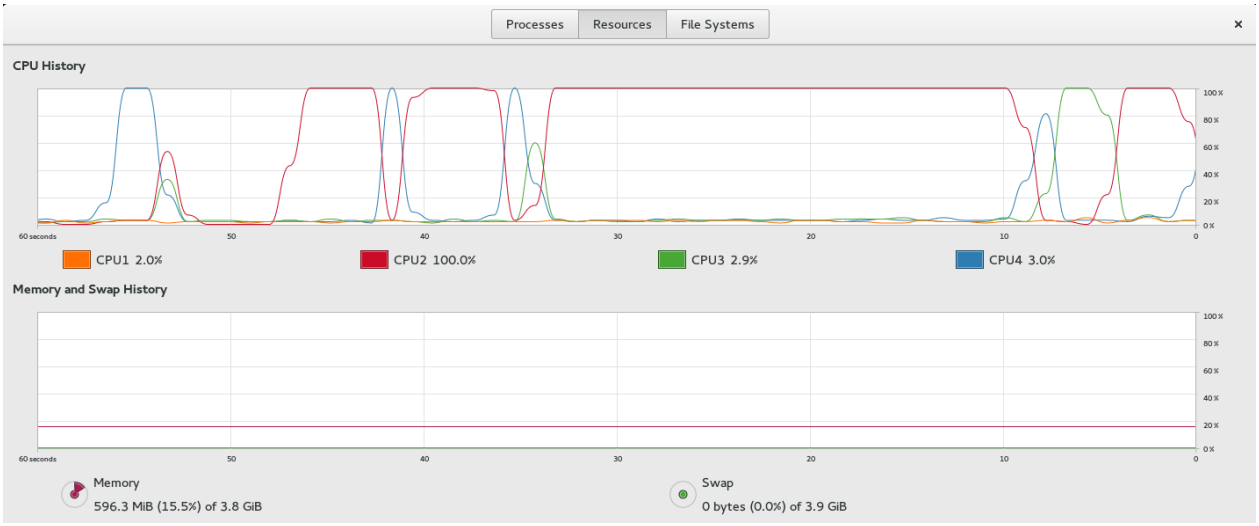

Figura 6. Uso de CPU durante la ejecución serial de los canales termohidráulicos

4. CONCLUSIONES

La programación en paralelo permite mejorar el aprovechamiento de las capacidades de las computadoras con microprocesadores multinúcleo. En sistemas Linux, mediante PVM se pueden desarrollar aplicaciones que permitan reducir considerablemente el tiempo de ejecución de cálculos numéricos en aplicaciones programadas en C o Fortran.

Para un caso de estudio de 36 canales termohidráulicos, una aplicación programada en paralelo corriendo en un equipo con 2 núcleos tarda entre el 54 y 56% del tiempo que demora en correr el mismo problema en el mismo equipo pero en su versión serie tradicional. Similarmente, en un equipo con 4 núcleos la ejecución toma entre el 22 y el 33% del tiempo demorado por la versión en serie.

Para el caso particular del estudio seleccionado en el presente documento, la programación en paralelo demuestra que se logran mejoras considerables en el tiempo de cómputo, y en general las aplicaciones en los que el tiempo de cómputo sea relevante se pueden beneficiar de la aplicación de la programación en paralelo.

Finalmente, los tiempos obtenidos están basados en un problema numérico, con poco acceso a disco, y donde los núcleos se aplican totalmente en resolver sus tareas sin necesidad de coordinarse para el uso de recursos compartidos. En aplicaciones donde sea necesario el uso intenso de un recurso compartido, no se logran los porcentajes de ahorro de tiempo obtenidos en el modelo trabajado como ejemplo en este documento, pero sí se logra mejorar la rapidez de manera significativa.

REFERENCIAS

1. Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, Vaidy Sunderam, *PVM: Parallel Virtual Machine. A User´s Guide and Tutorial for Networkes Parallel Computing*, Massachusetts Institute of Technology, USA (1994).

2. Marc Snir, Steve W. Otto, Steven Huss-Lederman, David W. Walker, Jack Dongarra, *MPI The Complete Reference*, Second Edition, MIT Press, USA (1996).

3. Miguel Ceceñas, Rina M. Campos, "Pruebas de Acoplamiento de Canales Paralelos a Cinética Neutrónica Modal" *XVIII Congreso Anual de la SNM*, Cancún, Quintana Roo, del 1 al 5 de Julio 2007.

4. T. Lefvert, "OECD/NEA BWR stability benchmark, final specifications", NEA/NSC/DOC(94)13. (1994).

5. S. Sahni, G. Vairaktarakis, "The master-slave paradigm in parallel computer and industrial settings" *Journal of Global Optimization*, **9**, p. 357-377 (1996).