Integración de elementos visuales y animaciones en las prácticas de programación paralela

José Miguel Mantas Daniel Guerrero Sergio Rodríguez
Departamento de Lenguajes y Sistemas Informáticos
Univ. Granada. Daniel Saucedo s/n, 18071 Granada
jmmantas@ugr.es danigrmartinez@gmail.com lumley256@gmail.com

Resumen

Dada la importancia actual del procesamiento paralelo y su complejidad, este trabajo pretende integrar el elemento visual y la animación dentro de la enseñanza y aprendizaje de la programación paralela. Con dicha orientación, se presentan las principales aportaciones de un proyecto de innovación docente que extiende el material web existente en las prácticas de una asignatura de programación paralela, dotándolo de elementos visuales dinámicos y cierto grado de interactividad para favorecer el aprendizaje. Asimismo, se introduce en las prácticas, el uso de herramientas de visualización de procesos y trazado gráfico de la ejecución que facilitan la comprensión y depuración de los programas paralelos. Las ideas presentadas pueden ser fácilmente aplicables a cualquier asignatura de programación que integre contenidos de concurrencia.

Palabras clave

Programación paralela, Visualización y animación de algoritmos paralelos, análisis y depuración de programas paralelos, Paso de mensajes, MPI.

1. Introducción

La programación paralela ha cobrado una gran importancia en los últimos años debido a la difusión y uso extendido de los procesadores multinúcleo, la computación en red y los procesadores gráficos programables. Dado que hoy en día, casi todos los sistemas de cómputo son paralelos, se le debe prestar una mayor atención a la enseñanza de la programación paralela y los algoritmos paralelos en los grados de Ingeniería informática [4]. Por otro lado, la programación paralela es mucho más compleja que la programación secuencial y este hecho se agudiza para el caso, de gran importancia, de la programación de plataformas distribuidas usando paso de mensajes [2, 14]. La dificultad de la programación paralela se debe principalmente al comportamiento dinámico de los programas, en los que múltiples entidades de ejecución (procesos o hebras) interactúan entre sí para lograr un propósito común. La sincronización y comunicación entre estas entidades hace que la comprensión del funcionamiento de los programas paralelos y la detección de errores sean especialmente complicados.

Es un hecho aceptado que la representación gráfica favorece la comprensión de sistemas complejos, ya que el sistema visual humano está más preparado para asimilar información en formato gráfico que en formato textual. En particular, para el aprendizaje de la programación y la algoritmia, el uso de enfoques basados en la visualización y animación del comportamiento han mostrado ser muy efectivos [9, 12]. Dada la mayor complejidad que exhiben los algoritmos paralelos, la adopción de enfoques de enseñanza basados en la visualización y animación [6] resulta aún más necesaria en este ámbito que en el ámbito de la programación secuencial. La comprensión de los algoritmos paralelos puede mejorarse tremendamente usando representaciones gráficas animadas, como se pone de manifiesto en algunos trabajos previos [8, 13]. Estas representaciones abstraen los aspectos fundamentales del algoritmo (forma en que se ha descompuesto el problema, sincronización entre las unidades de trabajo, transferencia de datos, etc.) y los presentan de una forma amigable al estudiante, permitiéndole captar más fácilmente el proceso de resolución paralela del problema.

En este trabajo, se presentan las principales aportaciones de un proyecto de innovación para mejorar el material web docente existente en las prácticas de una asignatura de programación paralela, dotándolo de elementos visuales dinámicos y cierto grado de interactividad.

La asignatura objeto de mejora pertenece al segundo ciclo de la Ingeniería Informática y aborda el diseño, análisis e implementación de algoritmos paralelos y distribuidos. En las prácticas, los estu-

diantes aprenden a programar y analizar algoritmos paralelos y distribuidos usando la interfaz de paso de mensajes MPI [2, 14] y en menor medida programan algoritmos paralelos basados en memoria compartida usando directivas y funciones de OpenMP [2, 5]. Se plantean varios guiones de prácticas como material de prácticas. En algunos guiones, el alumno trabaja con un tutorial web que sirve de introducción a la notación particular a usar (MPI u OpenMP). En estos tutoriales, se plantean problemas sencillos de programación paralela (ordenación, cuadratura numérica, cálculos de diferencias finitas, etc.) para que el alumno intente resolverlos, y más adelante se muestra el programa solución. El resto de guiones plantean proyectos de implementación paralela de pequeña envergadura (por ejemplo, implementación paralela de algoritmos de grafos). Estos guiones describen de forma esquemática un algoritmo paralelo para resolver el problema, y el alumno debe completar el desarrollo del proyecto incluyendo el análisis del rendimiento de su implementación paralela.

A lo largo de varios cursos, se han ido detectando problemas de aprendizaje en las prácticas, debido tanto a la dificultad para entender las funciones de paso de mensajes y la dinámica de los algoritmos distribuidos, como al arduo proceso de detección y corrección de errores en los programas paralelos. Para mitigar estos problemas, se propuso el proyecto de innovación docente cuyas aportaciones se presentan en este trabajo.

La sección 2 presenta el enfoque general seguido para mejorar las prácticas de programación paralela, mientras que los aspectos más relevantes de la implementación particular que se ha hecho de las ideas generales se describirá en la sección 3.

2. Enfoque general para introducir elementos visuales en asignaturas relacionadas con la programación paralela

El enfoque que se ha seguido para mejorar las prácticas (véase la Figura 1) está basado en las tres líneas de actuación que se presentan a continuación.

- Creación en la web de tutoriales, ayudas y guiones de prácticas que incluyan gráficos y animaciones interactivas. Estas animaciones describen los pasos principales de los algoritmos paralelos y la funcionalidad de las funciones subyacentes.
- Fomentar y apoyar en las prácticas el uso de herramientas de análisis y visualización de procesos que permitan comprender la interacción entre procesos en programas distribuidos.
- Adoptar el uso en prácticas de un entorno de desarrollo que integre una herramienta de depuración paralela con una alta componente visual.

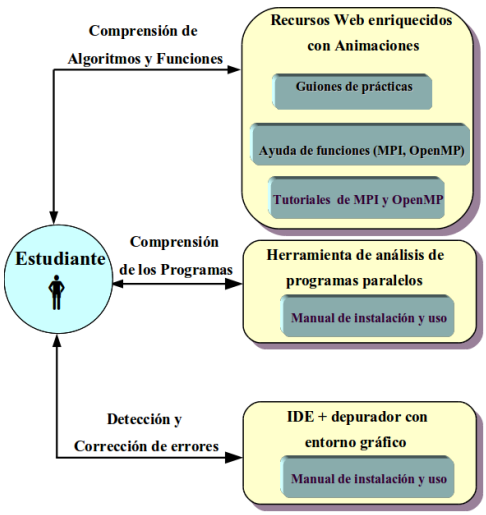

Figura 1: Visión general del enfoque propuesto

2.1. Inserción de Animaciones en Tutoriales y guiones de prácticas

Entender cómo funciona un algoritmo puede ser complicado, si además este debe funcionar parcial o totalmente de forma distribuida, la situación se vuelve aún más difícil. Generalmente el problema se encuentra asociado a la falta de entendimiento sobre qué parte del trabajo realiza cada proceso participante y, especialmente, en qué puntos se comparten los datos y qué datos son estrictamente necesarios para el correcto funcionamiento del algoritmo. En ocasiones, dichos datos se transmiten usando operaciones de comunicación colectivas (transferencia de uno a muchos, reparto de uno a muchos, recolección de muchos a uno, etc.) pensadas específicamente para simplificar la programación sin mermar el rendimiento del programa. Explicar este tipo de problemas claramente usando sólo un formato tex-

tual es complicado y no asegura el adecuado aprendizaje del estudiante.

Desde nuestro punto de vista, un desarrollador nuevo en el mundo del paralelismo carece de una estructura firme sobre la que recrear el funcionamiento de algoritmos distribuidos. Para solventar las dificultades de comprensión planteadas, recurrimos al dicho "más vale una imagen que mil palabras", y si son imágenes en movimiento que describan la naturaleza dinámica de un algoritmo paralelo, mejor.

En el enfoque propuesto se combinan dos aproximaciones diferentes, usando imágenes estáticas introductorias junto con animaciones para dar una descripción más detallada. La imagen representativa puede describir de forma abstracta el funcionamiento del algoritmo u operación relacionándolo con un entorno cotidiano. Usar una imagen de estas características permite reducir el tiempo dedicado al aprendizaje, y además, una vez aprendida la finalidad, el usuario tan solo deberá contemplar de nuevo la imagen para recordarla en un futuro.

Las animaciones, en cambio, tienen como finalidad mostrar los puntos clave en el tiempo de cada algoritmo u operación, así como los datos más relevantes que participan. Para facilitar el aprendizaje a largo plazo, todas las animaciones deberían usar el mismo modelo. De de esta forma un usuario inexperto, tras visualizar unas pocas animaciones, ya se encontraría familiarizado con los conceptos básicos de cada una de ellas y esto también le ayudaría para entender situaciones futuras.

Un ejemplo sencillo de explicación para el modelo gráfico es el caso de una operación de difusión de datos entre procesos. En esta operación, un único proceso envía exactamente los mismos datos a todos los procesos de un grupo determinado. La Figura 2 presenta una imagen que podría representar esta operación y la Figura 3 muestra algunos fragmentos de una animación descriptiva de la misma.

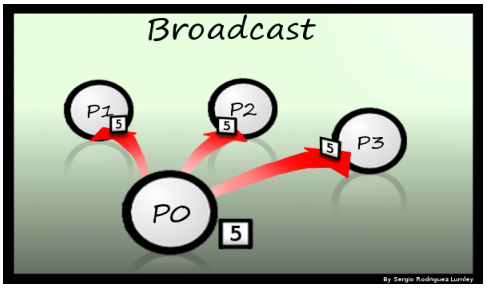

Figura 2: Imagen representativa de una difusión

2.2. Herramientas de visualización de procesos en programas basados en paso de mensajes

Cuando nos proponemos validar, analizar o comprender un programa paralelo, la tarea se vuelve más complicada que en un programa secuencial. Este análisis no es tan simple como hacer un seguimiento de la traza de una ejecución secuencial. En el caso de un programa paralelo de paso de mensajes, la existencia de múltiples procesos interactuando entre sí con operaciones de transferencia de datos, algunas de ellas bastante complejas, hace que analizar dichos programas sea una tarea ardua.

Una forma de abordar el análisis de programas basados en paso de mensajes consiste en usar herramientas de análisis del rendimiento de programas distribuidos [10, 15]. Entre otras posibilidades, estas herramientas permiten generar la información necesaria durante la ejecución de los programas para poder procesarla e interpretarla después de forma gráfica. Para visualizar la información recopilada por la herramienta, se hace uso de aplicaciones específicas para visualizar el comportamiento de un programa distribuido [3, 17]. Estas aplicaciones generan representaciones gráficas de los procesos donde las flechas denotan el paso de mensajes.

Los análisis visuales que permiten este tipo de herramientas favorecen la comprensión de los programas paralelos a los estudiantes que se inician en esta materia.

2.3. Uso de entornos de desarrollo con extensiones para depuración multiproceso

Para cualquier principiante en programación, disponer de la ayuda de un Entorno de Desarrollo Integrado (IDE) sobre el que poder trabajar de forma centralizada es una gran ventaja. Con estas herramientas a disposición del programador, el desarrollo de programas paralelos se facilita sustancialmente. El usuario puede dedicarse enteramente al desarrollo de aplicaciones sin preocuparse por detalles sobre compilación, dependencias y, en algunos casos, pueden recibir ayuda sobre errores sintácticos. A la hora de encontrar errores en la ejecución, los depu-

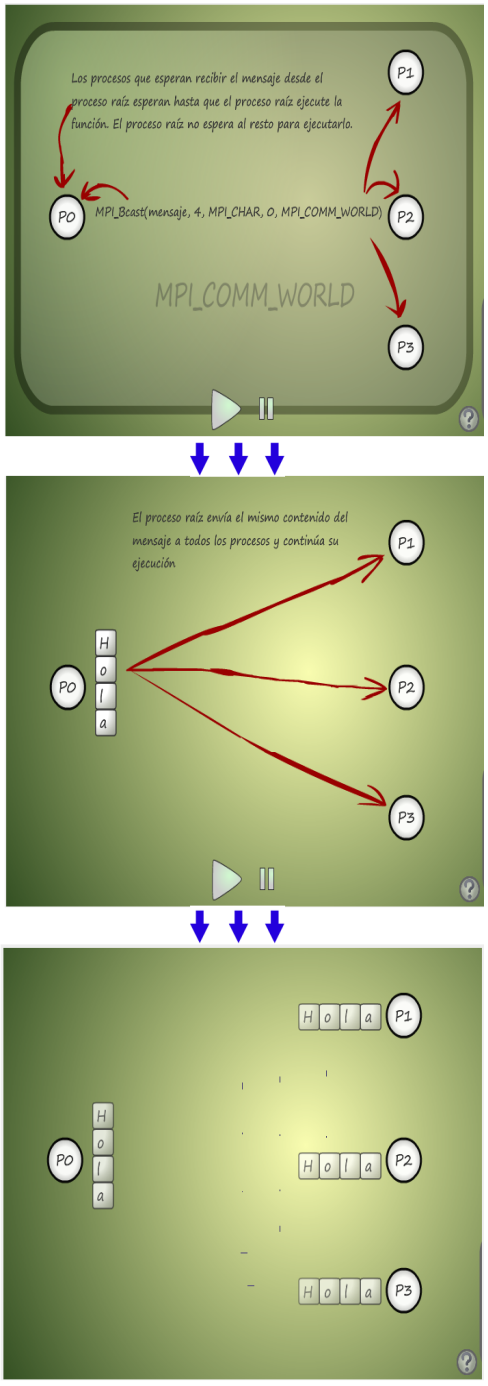

Figura 3: Animación para la difusión de uno a muchos

radores se convierten en una herramienta primordial, y cuando hablamos de depurar programas paralelos, más aún. Es muy común entre los programadores utilizar la salida por consola como método de depuración simple. Sin embargo esto no es lo más aconsejable en programas paralelos, puesto que intervienen múltiples procesos o hebras que avanzan de forma simultánea. Sin embargo, la mayoría de los depuradores tienen un problema añadido para principiantes, ya que suelen funcionar por consola y el usuario debe aprender a manejarlo. Por consiguiente, para facilitar lo máximo posible la tarea de depuración de programas paralelos, disponer de un depurador con representación gráfica es esencial.

Para que entorno de desarrollo y el depurador integrado sean útiles para nuestros propósitos, el estudiante debe poder visualizar algún tipo de representación gráfica de las unidades de ejecución, ya sean hebras y/o procesos, durante el trazado de la ejecución del programa paralelo, pudiendo consultar el estado de cada unidad de forma independiente en diferentes puntos de la ejecución.

3. Implementación particular en una plataforma web de una asignatura de programación paralela

El enfoque propuesto en la sección anterior ha sido implementado dentro de la página web http://lsi.ugr.es/ jmantas/pdp, dirigida a alumnos con un nivel considerable de programación secuencial, pero con poca experiencia en programación paralela. La web se encuentra centrada principalmente en la enseñanza práctica de la programación paralela mediante los métodos anteriormente descritos, incluyendo los siguientes recursos

- Animaciones dentro de los guiones prácticos.
- Un completo tutorial de MPI que describe los problemas planteados con ayuda de animaciones, así como un tutorial de OpenMP.
- Documentación web sobre MPI y OpenMP necesaria para abordar las prácticas, incluyendo animaciones para describir funciones de MPI.
- Guías de instalación en Linux y manuales de uso de las principales herramientas usadas. Dentro de las herramientas se incluyen:

  1. Una Herramienta de análisis y visualización de programas paralelos, denominada

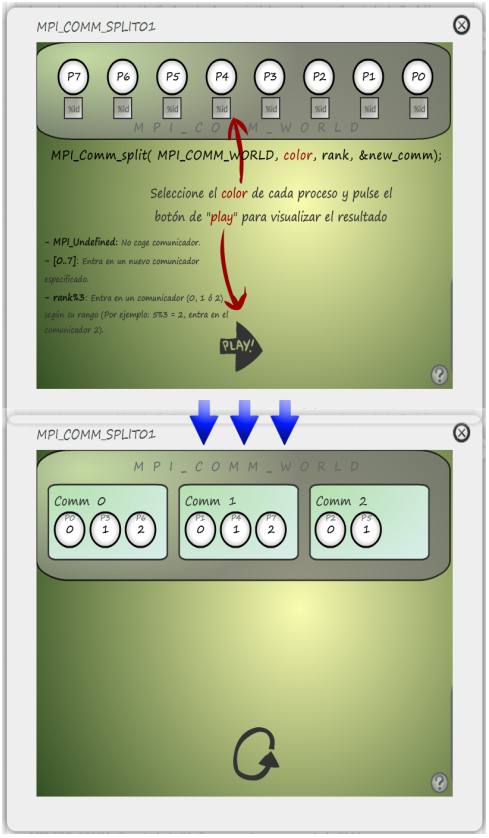

Figura 4: Ejemplo de animación Flash interactiva

TAU [15].

2. El IDE *Eclipse* y su depurador para programas paralelos.

3.1. Diseño de las animaciones

Siguiendo la filosofía propuesta, todas las animaciones se introducen mediante una imagen que ilustra del funcionamiento descrito (véase un ejemplo en la Figura 2). Si el usuario necesita aún más información, pulsando sobre la imagen se desplegará la animación (véase la Figura 3). Esta forma de funcionar permite que un mayor número de dispositivos vean la web correctamente y también permite evitar la carga de la animación cuando no se dispone de conexiones rápidas. Para el diseño de animaciones hemos recurrido al ampliamente utilizado *Flash* [1, 11], lo que permite que sean visualizadas por prácticamente cualquier navegador actual, además de poder dotar a las animaciones de cierto grado de interactividad. Todas las imágenes y animaciones siguen el mismo modelo para lograr una mayor adaptación del usuario y permitirle centrarse antes en el contenido que en el funcionamiento de la propia animación. Para el caso de las animaciones, siempre siguen la misma secuencia: una secuencia de introducción animada, presentación del cuadro de ayuda y ocultación del mismo, y reproducción del resto de la animación (véase la Figura 4).

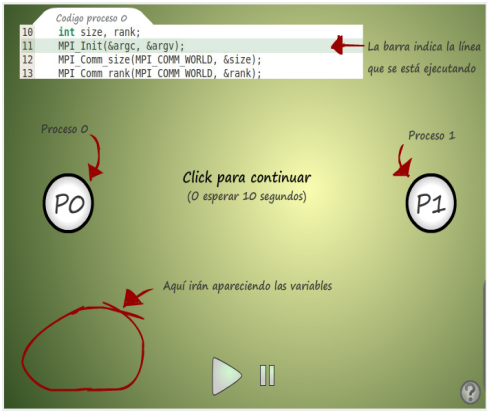

Figura 5: Ejemplo de animación Flash cinemática

Cuando se presenta texto en pantalla, se muestran controles del flujo de la presentación y las opciones disponibles son: pulsar sobre la pantalla, para continuar con la siguiente secuencia; pulsar sobre pausa para detener la animación; o no hacer nada, en cuyo caso la animación continuará cuando pase tiempo suficiente (véase la Figura 5).

Las animaciones para las funciones muestran su funcionamiento en un orden temporal, destacando el uso de los parámetros (véase la Figura 3). Las animaciones para los tutoriales muestran qué debe hacer el programa que resuelve el problema planteado y algunos detalles sobre cómo lograrlo. Como muestra, la Figura 6 esboza un algoritmo distribuido de ordenación Mergesort. En las animaciones para prácticas, se explica cuál es el comportamiento que se debe lograr, utilizando representaciones similares a las ya presentadas, para que el usuario pueda reconocerlas fácilmente. Así, en la Figura 7 se muestra el marco de una animación que describe una implementación paralela del algoritmo de Floyd para

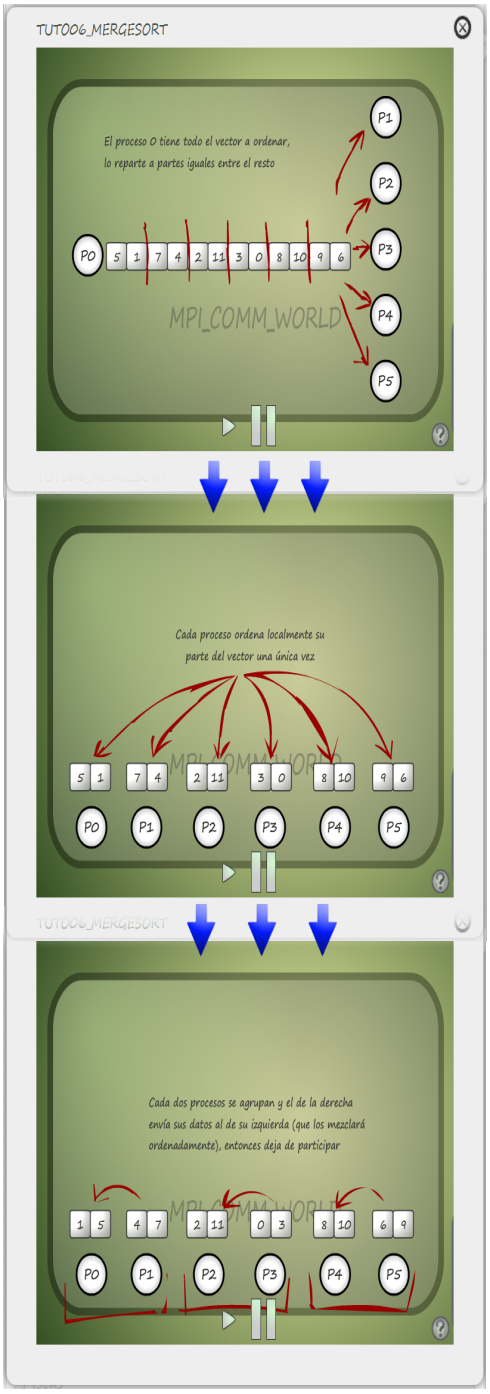

Figura 6: Ejemplo de animación en tutorial

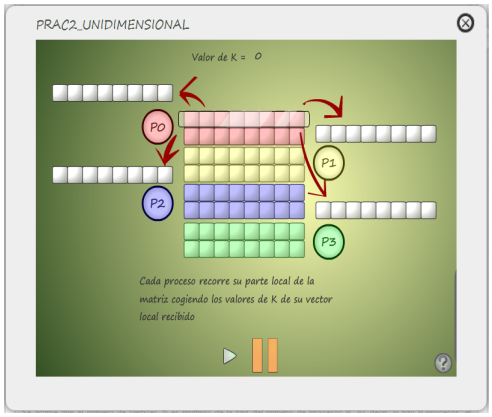

Figura 7: Ejemplo de animación en guión de prácticas

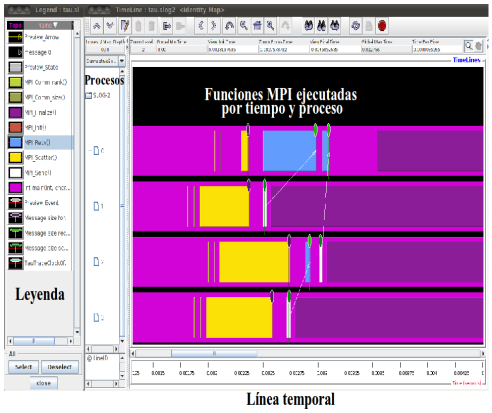

Figura 8: Salida de la herramienta TAU

buscar los caminos más cortos en un grafo.

3.2. Integración de una herramienta de análisis de programas de paso de mensajes

Para facilitar su uso a los estudiantes, se ha optado por implantar un software gratuito de análisis de programas MPI denominado TAU (Tuning and Analysis Utilities). TAU es un conjunto de utilidades que nos permiten analizar programas paralelos escritos en lenguaje Fortran, C, C++, Java y Python. TAU nos permite hacer un análisis en profundidad de los programas paralelos de forma parecida a los analizadores convencionales, además de realizar un análisis del trazado de las comunicaciones entre procesos. Los resultados de dichos análisis pueden ser

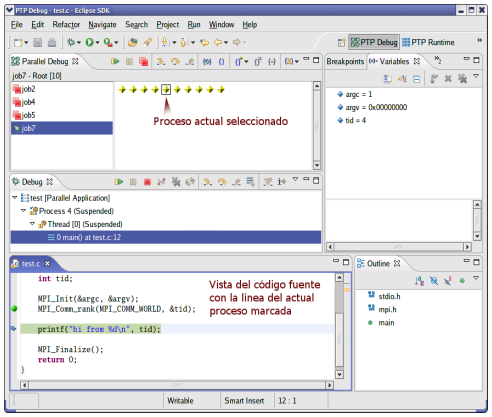

Figura 9: Depuración con Eclipse y PTP

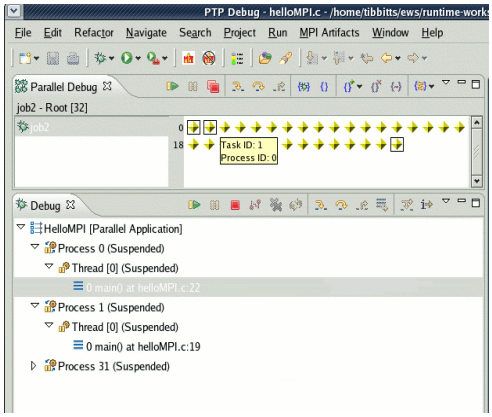

Figura 10: Control de hebras con un IDE

visualizados de forma gráfica gracias a herramientas como *Paraprof* [17], destinada a realizar estadísticas del análisis, y *Jumpshot* [3] que muestra el trazado de las funciones de comunicación entre los procesos en una línea temporal.

El uso de TAU para visualización es sencillo. Se usa un script específico para compilación y enlace del programa MPI que genera un ejecutable extendido. Al lanzar este ejecutable, además de la salida normal del programa, se guardan datos en ficheros durante su ejecución. Estos ficheros pueden ser unificados y visualizados con *Jumpshot*. La Figura 8 muestra la visualización obtenida para un programa MPI sencillo. En la parte inferior se muestra una línea temporal y los diferentes procesos aparecen como barras horizontales divididas en franjas de diferentes colores que pueden hacer referencia a la ejecución de diferentes funciones de MPI (tal como indica la leyenda que aparece en el lado izquierdo). Los puntos de comunicación entre procesos se indican claramente con elipses y flechas que unen diferentes instantes de tiempo en la representación gráfica de los procesos.

3.3. IDE con depurador paralelo

Se ha optado por un IDE lo suficientemente sencillo de usar, extensible y gratuito, por lo cual el popular IDE *Eclipse* [7] cumplía las características buscadas para nuestros propósitos, además de ser de código abierto. A través del sistema de extensiones del que dispone, Eclipse cuenta en su repertorio con la mejor de las opciones para un desarrollador de programas distribuidos, un depurador gratuito, libre, gráfico y con la capacidad de controlar programas distribuidos, hablamos de la extensión *Parallel Tools Platform* (*PTP*) [16]. Esta herramienta permite depurar programas divididos en varios procesos. En tiempo de ejecución, el programa permite seleccionar cualquiera de los procesos y controlar su ejecución mostrando el depurador en qué línea de código se encuentra el proceso, pausar el resto de procesos o hacer que avancen hasta el mismo punto que el proceso seleccionado. La Figura 9 muestra el IDE durante el proceso de depuración, distinguiéndose el proceso seleccionado y la línea de código que está ejecutando.

La herramienta es incluso capaz de controlar hebras dentro de cada proceso, por lo que está indicada también para trazar programas paralelos que usan OpenMP, como se muestra en la Figura 10.

3.4. Evaluación

Aunque no se ha tenido tiempo de realizar una evaluación rigurosa, se ha pasado a los alumnos un cuestionario para detectar posibles deficiencias del sistema implantado. La Tabla 3.4 resume los resultados obtenidos (la máxima valoración es 5) y refleja en general una buena aceptación del uso de las animaciones. La implantación de las herramientas de análisis y depuración ha sido parcial ya que se pretende ir introduciendo los nuevos elementos paulatinamente en dos cursos académicos. No obstante, los estudiantes que han optado por usar Eclipse con el

Pregunta	Med.
Calidad y efectividad de las guías de ayuda	4,2
Comprensión del texto en ayuda de MPI	4,2
Efectividad de imágenes en la ayuda de MPI	4,4
Efectividad de animaciones en la ayuda de MPI	4,5
Comprensión del texto en la ayuda de OpenMP	3,7
Comprensión del texto en tutoriales y prácticas	4,2
Efectividad de imágenes en tutoriales y prácticas	4,4
Efectividad de animaciones en tutoriales y prácticas	4,5

Cuadro 1: Resumen de resultados del cuestionario

plugin PTP se han mostrado muy satisfechos con los beneficios obtenidos tanto en la detección de errores como en la comprensión de la ejecución paralela.

4. Conclusiones y Trabajos Futuros

En este trabajo se proponen tres tipos de actuaciones para mejorar el aprendizaje de la programación paralela. Estas ideas están inspiradas en enfoques basados en la animación y visualización y se han implementado en la página web de una asignatura de programación paralela. La combinación de animaciones para describir el comportamiento de los algoritmos paralelos, junto con el uso de herramientas de análisis, visualización y depuración gráfica de programas paralelos, se muestra como una via adecuada para mejorar el aprendizaje de los aspectos más complejos de la programación paralela. Además estas ideas se pueden implantar fácilmente en asignaturas de fundamentos de programación y algoritmia en las que se decida impartir contenidos sobre computación y algoritmos paralelos.

Actualmente, se está trabajando en la extensión del trabajo con las siguientes mejoras:

- Insertar animaciones para enriquecer el material web dedicado a OpenMP y extender el material web con recursos para el aprendizaje de la programación de GPUs.
- Desarrollar una aplicación web interactiva que permita la edición de imágenes y animaciones, siguiendo las líneas guía ya introducidas.

Agradecimientos

El trabajo se ha financiado con los proyectos 09-162 de la Univ. de Granada (Innovación Docente) y MTM2008-06349-C03-03 (MEC).

Referencias

[1] Adobe Flash CS4 Professional. www.adobe.com/es/products/flash.

[2] Almeida F., Giménez D., Mantas J. M., Vidal A. M. *Introducción a la programación paralela*. Paraninfo Cengage Learning, 2008.

[3] Bell R., Malony A. D., Shende S. *ParaProf: A Portable, Extensible, and Scalable Tool for Parallel Performance Profile Analysis*. Europar 2003, LNCS 2790 pp.17-26, 2003.

[4] Giménez D., Almeida F., Mantas J. M., Vidal A. M. *Sobre la situación del paralelismo y la programación paralela en los Grados de Ingeniería Informática*. ReVisión, Vol 3, N 1, 2010.

[5] Chapman B., Jost G., van der Pas R. *Using OpenMP. Portable Shared Memory Parallel Programming*. The MIT Press, 2007.

[6] Gómez-Albarrán M. *Una revisión de métodos pedagógicos innovadores para la enseñanza de la programación*. JENUI 2003 pp.: 363–370.

[7] IDE Eclipse. http://www.eclipse.org/.

[8] Johnson D., Kotz D., Makedon F. *Teaching Parallel Computing to Freshmen*. Conference on Parallel Computing for Undergraduates 1994.

[9] Mandow L., Villalba F., Coego J. *Uso de animaciones para la enseñanza de algoritmos de búsqueda en Inteligencia Artificial*. JENUI 2010 pp.: 461–468.

[10] MPE. www.mcs.anl.gov.

[11] Mook C. *ActionScript : The Definitive Guide*. O' Reilly Media 2001.

[12] Pérez A.., Velázquez J. A. *Animación automatizada de técnicas de diseño de algoritmos*. JENUI 2009 pp.: 115–122.

[13] Rantakokko J. *Algorithm Visualization through Animation and Role Plays*. Proc. of 3rd Progr. Visualization Workshop, 2004. pp.: 76–81.

[14] Snir M., W. Gropp. *MPI. The Complete Reference. 2nd edition*. The MIT Press, 1998.

[15] TAU. Tuning and Analysis Utilities. www.cs.uoregon.edu/research/tau.

[16] Watson G., Rasmussen C., Tibbitts. B. *An integrated approach to improving the parallel application development process*. IEEE ISPDP 2009, pp.1-8, 2009.

[17] Wu C. E., et. al. *From Trace Generation to Visualization: A Performance Framework for Distributed Parallel Systems*. Proc. of SC2000.