ParallelJ: Entorno de desarrollo y simulación de programas paralelos

José Manuel García Alonso, José Javier Berrocal Olmeda,
Juan Manuel Murillo Rodríguez
Dpto. de ingeniería de sistemas informáticos y telemáticos
Universidad de Extremadura
Escuela Politécnica, Avenida de la Universidad s/n, 10071 Cáceres
jgaralo@unex.es, jberolm@unex.es, juanmamu@unex.es

Resumen

La docencia de materias relacionadas con la algoritmia paralela presenta serias dificultades. Por una parte no siempre es posible disponer de multiprocesadores o multicomputadores. Además, cuando se dispone de ellos la dificultad para su configuración distrae la atención del alumno restándosela a la algoritmia. Por otra parte las herramientas de simulación son muy limitadas y se encuentran claramente desfasadas.

En este artículo se presenta ParallelJ, una herramienta que ayuda a solventar este tipo de problemas. Los beneficios de la utilización de ParallelJ han podido constatarse en la docencia de diversas asignaturas.

1. Introducción

Una de las materias curriculares de los titulados en informática es la relativa a la computación paralela. Dentro de este ámbito se incluyen conocimientos de algorítmica, modelos de programación, arquitecturas paralelas, mecanismos de comunicación y sincronización de procesos, etc.

Usualmente, el aprendizaje de materias relativas a la computación se apoya en herramientas para la realización de prácticas. Un gran conjunto de estas herramientas lo componen las plataformas de programación basadas en diferentes lenguajes. La utilidad de estas es doble; mientras que por un lado ofrecen al alumno una clara visualización del flujo de computación, por otro afianzan al alumno en los conocimientos adquiridos. La necesidad de estas herramientas se hace vital cuando se trata de materias relativas a la computación paralela que exigen al alumno un mayor esfuerzo para tratar varios flujos de computación que corren simultáneamente en paralelo.

Sin embargo, las herramientas que se encuentran a disposición de alumnos y profesores para la realización de prácticas de computación paralela plantean algunos problemas. En primer lugar las soluciones tecnológicas necesarias para experimentar en algunas de las ramas descritas son excesivamente costosas (es necesario disponer al menos de dos ordenadores). Este hecho hace difícil la utilización de dichas tecnologías de forma doméstica. En segundo lugar las dificultades tecnológicas que plantea la configuración de las plataformas para la computación paralela eclipsan los conocimientos de la algorítmica, distrayendo la atención del alumno.

En este artículo se presenta ParallelJ, una herramienta que permite al alumno diseñar, programar, ejecutar, depurar y comparar algoritmos paralelos de forma simulada en su propio ordenador personal. Esta herramienta proporciona un entorno de desarrollo integrado con Eclipse. El lenguaje de programación propuesto es una extensión del lenguaje de programación Java con primitivas para creación y comunicación de procesos. Los resultados de la ejecución de los programas se ve ilustrada con gran número de informes y estadísticas sobre la distribución de procesos, la utilización de procesadores y aprovechamiento de la red de comunicaciones. Todo ello facilita notablemente al alumno el acceso a dificultades planteadas por estas materias.

En la siguiente sección se describirán los antecedentes que han motivado la necesidad de crear la herramienta ParallelJ. En la sección tres se describe en detalle las características del recurso docente presentado, prestando especial atención al

lenguaje utilizado por la herramienta, las capacidades de simulación de que dispone, las estadísticas que permite generar, su integración con un entorno de desarrollo y la licencia bajo la que se distribuye. El artículo se cierra con una revisión de los resultados obtenidos tras la implantación de la herramienta en varias asignaturas relacionadas con la computación paralela.

2. Antecedentes: Estado del arte

En general la docencia de la algoritmia paralela se divide en dos corrientes diferenciadas por el tipo de herramientas utilizadas para asistir a la docencia y que permiten a los alumnos probar los algoritmos diseñados. Una de estas corrientes se basa en la utilización de herramientas de paralelismo real como PVM [9], MPI [7] o RMI [10] y la otra en la utilización de lenguajes didácticos y simuladores como Superpascal y Multipascal.

La utilización de herramientas que permiten explotar una arquitectura paralela real cuenta con una gran ventaja frente al resto de las opciones disponibles. Los resultados obtenidos durante la realización de las pruebas de los distintos algoritmos son reales, puesto que estas pruebas se han ejecutado realmente sobre una arquitectura multiprocesador (o multicomputador) con todas sus características intrínsecas. Sin embargo esta ventaja no contrarresta los problemas que esta alternativa plantea para su uso docente. El problema más grande de este tipo de tecnologías, a la hora de su utilización como herramienta docente, reside en la necesidad de disponer de la arquitectura multicomputador sobre la que se desee probar el algoritmo diseñado, que si bien puede no suponer un problema en un centro académico si que lo es en una casa particular. Sin obviar también que, aunque se disponga de una arquitectura de este tipo, su topología es estática lo que limita en gran medida la variedad de técnicas y configuraciones que pueden ser probadas por el alumno. Además, otro de los problemas importantes que presentan estas tecnologías es la complejidad de uso y mantenimiento de las mismas, que puede derivar la atención del alumno de la algorítmica al aprendizaje de las particularidades de una plataforma.

Para solventar estos problemas en la docencia de la algoritmia paralela se plantea el uso de soluciones como Superpascal [3], un lenguaje de programación concebido para la transmisión (publicación) de algoritmos paralelos en el ámbito científico y docente. Sin embargo este tipo de soluciones no proporciona toda la funcionalidad necesaria para la docencia puesto que no es posible observar los resultados de la ejecución de los algoritmos escritos en este lenguaje. Los algoritmos diseñados con Superpascal deben traducirse a la sintaxis concreta de algún sistema para poder ser probados. Esto limita su posible utilidad como herramienta docente.

Una herramienta más avanzada es Multipascal [6] que permite al usuario abstraerse de las complejidades reales propias de las arquitecturas paralelas ofreciendo una gran flexibilidad para escribir y probar este tipo de algoritmos. Multipascal es un simulador que permite escribir algoritmos paralelos y simular su ejecución en una gran variedad arquitecturas sobre un único ordenador, permitiendo además obtener una serie de estadísticas de la ejecución muy útiles para la docencia. Mediante la utilización de esta herramienta es posible, por lo tanto, llevar a cabo un aprendizaje de las técnicas de programación paralela y comparar el funcionamiento de distintos algoritmos paralelos sin realizar un gran desembolso para tener al alcance una arquitectura paralela, eludiendo además las tareas de configuración de ésta.

Sin embargo esta herramienta aparece totalmente obsoleta. En primer lugar el lenguaje básico que se utiliza es similar, en cuanto a su sintaxis, a Pascal. Este lenguaje de programación fue desarrollado alrededor de 1970 por Niklaus Wirth alcanzando su apogeo entre los años 80 y el principio de los 90. Es por lo tanto bastante improbable que algún alumno tenga conocimientos previos o vaya a utilizar el conocimiento adquirido de Pascal a lo largo de su carrera profesional.

Por otra parte, a diferencia de los IDE de desarrollo actuales, el propio entorno carece de un editor específico. Por lo tanto, la escritura de los ficheros que sirven de fuente para este programa debe llevarse a cabo en un editor ajeno a la herramienta que acarrea, además de la molestia de obligar al usuario a cambiar de entorno

constantemente, la dificultad de no prestar muchas facilidades para la edición del código fuente.

Además de esta dificultad, quien desee utilizar Multipascal se encontrará con una serie de problemas.

1. Tanto el intérprete como el simulador de programas son herramientas de linea de comandos con pocas opciones disponibles.
2. La generación de estadisticas sobre la ejecución es muy limitada.
3. Presenta muy pocas facilidades de cara a la depuración de programas.
4. Las posibilidades de simulación son bastante limitadas debido a la incapacidad de gestionar arquitecturas de más de unos pocos cientos de procesadores.

Finalmente, aunque algunos de estos problemas podrían solventarse realizando algunas adaptaciones a Multipascal, esto no es posible debido a que el código fuente del sistema no está disponible bajo una licencia de software libre.

Todos estos antecedentes hacen patente la necesidad de herramientas docentes que permitan la programación, ejecución simulada y depuración de algoritmos paralelos de forma cómoda, moderna, asequible y proporcionando informes relativos a su eficiencia, rendimiento, utilización de la arquitectura, etc. Justo con esta motivación nace ParallelJ.

3. ParallelJ

La herramienta ParallelJ pretende cubrir la necesidad de un sistema que abarque todo el proceso de desarrollo de un algoritmo paralelo, desde la escritura del código hasta la generación de estadísticas de la ejecución. Así, la herramienta consiste en un entorno de desarrollo integrado que permite escribir, ejecutar, depurar y obtener estadísticas de la ejecución de algoritmos paralelos sobre arquitecturas distribuidas con diferentes topologías de conexión simuladas sobre un único procesador. A continuación se describen las características más importantes del sistema.

3.1. Características del lenguaje

El lenguaje de programación de ParallelJ es uno de los elementos cruciales del sistema puesto que es el primer escollo al que deben enfrentarse los alumnos para poder construir sus algoritmos paralelos. Con el ánimo de permitir salvar al alumno la dificultad del lenguaje con el menor esfuerzo posible, ParallelJ se basa en Java [4] que es un lenguaje de amplia utilización en la industria y muy conocido por los alumnos.

El lenguaje de programación ParallelJ cuenta con todas las funciones básicas propias de un lenguaje de propósito general orientado a objetos como es Java. Así permite utilizar tipos primitivos básicos como *int*, *float*, *boolean*… y combinarlos mediante expresiones arbitrariamente complejas con operadores aritméticos, operadores lógicos paréntesis, etc. Se permite también al usuario utilizar objetos que encapsulan a estos tipos primitivos *Integer*, *Float*, *Boolean*… y se proporciona los métodos comunes para estos objetos como por ejemplo aquellos destinados a convertir el valor del objeto a una cadena de texto, obtener el tipo primitivo correspondiente al valor del objeto, etc. Además el programador puede definir sus propios objetos que encapsulan tantos atributos y métodos como sean necesarios. El lenguaje cuenta también con el conjunto de instrucciones habituales para el control del flujo de ejecución propias de Java. Para el control de condiciones se dispone de los bloques *if-else* y para la ejecución de bucles de las instrucciones *while* y *for*. Se permite también realizar las llamadas a métodos de cualquier tipo de objetos.

Además de todas estas características, comunes a la mayoría de los lenguajes de propósito general, el lenguaje ParallelJ dispone de algunas características especiales. Una de estas características es la utilización de vectores dinámicos en tamaño tanto para tipos primitivos como para objetos de cualquier tipo. De esta forma no es necesario fijar el tamaño de un vector en tiempo de compilación y el programador no debe preocuparse por posibles desboramientos o índices fuera de rango, pudiendo trabajar de forma similar a como se trabaja en el lenguaje Java con las colecciones.

El resto de características específicas del lenguaje ParallelJ son características propias de los lenguajes paralelos y que podemos clasificar en tres grupos.

En primer lugar el *grupo de instrucción para la creación de procesos*. Para el lanzamiento de nuevos procesos se dispone de dos primitivas, la primitiva *fork* para lanzar un único proceso y la

primitiva *forall* para lanzar un conjunto de procesos. Ambas primitivas reciben un método como parámetro que constituirá el código del proceso que se crea. La primitiva forall crea una serie de procesos gemelos todos ejecutando el mismo código. Un proceso puede esperar por la finalización de alguno de los procesos que creó utilizando la primitiva *join*.

El segundo es el *grupo de instrucciones para la comunicación de procesos*. La primitiva *send* para enviar un mensaje, la primitiva *receive* para recibir un mensaje y la primitiva *existsMessages* para comprobar la existencia de mensajes. Para que un proceso pueda enviar un mensaje a otro utilizando la primitiva *send* debe indicar el identificador del proceso destino. El contenido de los mensajes puede ser cualquier de los tipos primitivos del lenguaje o cualquier objeto y se puede realizar un filtrado de los mismos mediante etiquetas de texto. Estas etiquetas de texto pueden utilizarse también en las primitivas *receive* y *existsMessages* para filtrar la recepción de mensajes.

```
class          Proceso_Comunicacion
extends Process{
   void run(int pos){
        String msj;
        int parent;
   System.out.println("Esperando
mensaje");
        msj=receive("ini");
        System.out.println("Mensaje
recibido: "+msj);
        parent=getParent();
   send("msj",parent,"Respondiendo
al mensaje");
   }
}
```

Algoritmo 1. Ejemplo de un proceso en ParallelJ

Finalmente, el *grupo de instrucciones que permiten a un proceso obtener información sobre el entorno de ejecución*, como la primitiva *getId* para obtener el identificador de un proceso, la primitiva *getParent* para obtener el identificador del padre de un proceso, la primitiva *getChildren* para obtener los identificadores de los hijos de un proceso, la primitiva *getProcessor* para obtener el procesador sobre el que se está ejecutando un proceso, etc.

La suma de todas estas características convierte a ParallelJ en un lenguaje potente y flexible que permite la implementación de cualquier tipo de algoritmo paralelo.

3.2. Características de la simulación

Aunque el lenguaje es una de las principales características del sistema su utilidad para la docencia se ve drásticamente disminuida si no se cuenta con la capacidad de poder probar los algoritmos escritos. Por lo tanto la funcionalidad principal de ParallelJ es la simulación de algoritmos paralelos y es en este apartado donde la herramienta muestra toda su potencia y flexibilidad.

Desde ParallelJ es posible ejecutar o depurar cualquier algoritmo, configurando una amplia gama de características para ajustar la simulación al sistema real que fue elegido lo más fielmente posible. Es posible seleccionar la topología de la red de computadores sobre la que se va a simular la ejecución entre una gran variedad que incluye líneas, anillos, matrices bidimensionales y tridimensionales, hipercubos, toroides y fullconnect. Es posible modificar el número de procesadores que van a intervenir en la simulación en función de la topología de red elegida. Este punto requiere especial atención puesto que ParallelJ permite realizar simulaciones mucho más masivas que herramientas anteriores como Multipascal que no contaban con capacidad suficiente para simular más que unos pocos de cientos de procesadores. En ParallelJ es posible realizar simulaciones que utilizan varios miles de procesadores. Esto proporciona al alumno una perspectiva más amplia y a mayor escala del rendimiento de los algoritmos diseñados.

Otro de los aspectos más importantes y a la vez más flexibles de la simulación es el tratamiento de las comunicaciones y el comportamiento de la red. Es posible configurar estos parámetros de forma que la simulación se realiza sobre una red con unas características lo más similares posibles a la red real que se quiere emular. Así, se permite configurar el Retraso Básico de Comunicaciones o tiempo que tarda un mensaje en atravesar un enlace de la red, además, como en un sistema real, la simulación tiene en cuenta que estos enlaces se sobrecargan en función del tráfico que transcurra por los mismos modificándose el tiempo que tarda un mensaje en

recorrerlos en función del tráfico que soportan. Asímismo, la herramienta simula un enrutamiento dinámico en función de la sobrecarga de los enlaces de la red. También es posible elegir entre varias posibilidades de congestión de la red (ninguna, constante o aleatoria) así como la influencia de la misma en las comunicaciones de tal forma que sea posible realizar simulaciones que permitan comprobar el funcionamiento de los algoritmos sobre distintos estados de saturación de la red.

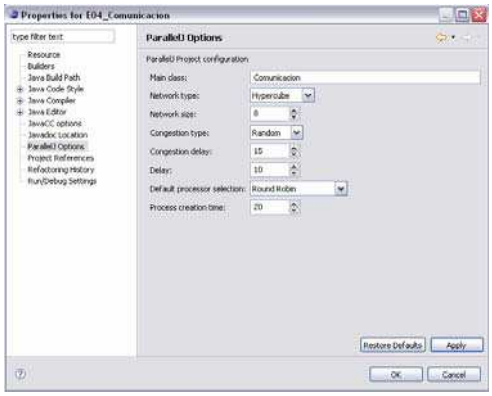

Figura 1. Ventana de configuración de los parámetros descritos

Además ParallelJ dispone de algunos parámetros más que permiten personalizar aun más la ejecución de los algoritmos para adaptar las condiciones de simulación a las que se consideren necesarias. Entre estos parámetros destacan la posibilidad de modificar el tiempo de creación de un proceso a partir de la imagen del mismo que se envía por la red y la posibilidad de elegir entre varias políticas de asignación de procesos a procesadores (aleatoria, *round robbin*, menor carga) cuando estos se lanzan sin indicar un procesador especifico de destino.

Todas estas propiedades hacen posible realizar simulaciones reproduciendo condiciones reales de un entorno de ejecución. Este hecho permite a los alumnos realizar pruebas exhaustivas del comportamiento de sus algoritmos en un amplio espectro de situaciones, comparar los resultados obtenidos y obtener conclusiones fiables sobre el funcionamiento de los mismos en un entorno real.

3.3. Estadísticas

Contar con un potente motor de simulación, como es el caso de ParallelJ, permite realizar una gran variedad de pruebas y obtener unos resultados fiables, sin embargo, esto puede no ser suficiente en el ámbito docente. En muchas ocasiones se hace necesaria la presencia de datos adicionales para que los resultados de las simulaciones puedan ser interpretados correctamente. Para solventar este problema ParallelJ proporciona la posibilidad de obtener un amplio rango estadísticas para cada simulación. Muchas de estas estadísticas se proporcionan en formato gráfico gracias a la herramienta JFreeChart [5]. A continuación se detallan las estadísticas que es posibles obtener tras cada simulación de un algoritmo en ParallelJ:

- ParallelJ.log: Un pequeño archivo de texto que contiene datos básicos de la ejecución tales como el tiempo de ejecución en paralelo, el tiempo correspondiente a la ejecución en secuencial, el speedup obtenido, el número de procesadores utilizados durante la ejecución…
- Mensajes por proceso: Una gráfica de barras que muestra el número de mensajes enviados por cada proceso así como sus destinatarios. Esta gráfica permite constatar el programador que el envío de mensajes se comportó tal y como el preveía.

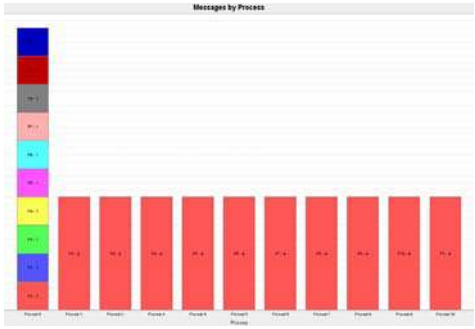

Figura 2. Ejemplo de mensajes por proceso para la ejecución de un algoritmo bajo el esquema token-ring

- Ruta de los mensajes: Archivo de texto en el que, para cada mensaje enviado durante la simulación, aparece el proceso origen, el proceso destino, la etiqueta utilizada en el mensaje y la ruta de procesadores seguida por

el mismo incluyendo los procesadores que ejecutan a los procesos origen y destino. A través de este informe el programador puede conocer cual es el tráfico real que el programa generó en la red en función de la sobrecarga generada por el algoritmo.

- Procesos por procesador: Una gráfica de barras que muestra el número de procesos que se ha ejecutado sobre cada procesador así como el porcentaje de utilización del mismo que corresponde a cada proceso.
- Estado del proceso: Gráfica que muestra la evolución de un proceso a lo largo de la ejecución, mostrando para cada momento el estado en que se encontraba el proceso: ejecutando, esperando por un mensaje, esperando por el procesador, etc.
- Perfil del procesador: Gráfica que muestra el perfil de utilización de un procesador, mostrando para una rodaja de tiempo, el porcentaje de utilización del procesador. El tamaño de la rodaja de tiempo puede ajustarse. Se genera una gráfica por procesador y una gráfica conjunta para todos los procesadores.
- Perfil por bloques: Gráfica bidimensional con el tiempo (dividido en rodajas) en el eje x y los diferentes procesadores en el eje y. La gráfica muestra, mediante un código de colores, la utilización de todos los procesadores a lo largo de la ejecución. El color verde indica una utilización entre el 0 y el 25%, el amarillo una utilización entre el 25 y el 50%, el naranja una utilización entre el 50 y el 75% y el rojo una utilización entre el 75 y el 100%. Esta gráfica resulta especialmente útil para hacerse una idea de la utilización en conjunto de los distintos procesadores a lo largo de la ejecución proporcionando una idea real de la explotación de la arquitectura paralela llevada a cabo por el algoritmo paralelo. Además, en muchas ocasiones, permite obtener información relevante sobre el esquema de comunicaciones o la arquitectura elegida, como es el caso de la figura 3.

Todos estos informes consiguen aportar claridad a lo que ha sucedido durante la simulación de forma que sea más asequible para los alumnos sacar conclusiones sobre los resultados obtenidos.

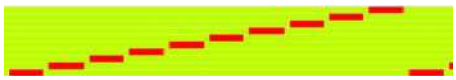

Figura 3. Ejemplo de perfil por bloques para la ejecución de un algoritmo bajo el esquema de token-ring

3.4. Integración con Eclipse

Los puntos descritos anteriormente conforman la base de la funcionalidad de ParallelJ, sin embargo todas estas características pierden gran parte de su utilidad si no se dispone de una interfaz lo suficientemente cómoda y sencilla que permita realizar las distintas tareas con facilidad.

Puesto que ParallelJ abarca todos los campos básicos de un entorno de desarrollo integrado se ha optado por integrar la aplicación con uno de los entornos de desarrollo más utilizados actualmente, Eclipse [1]. Con esta decisión se consiguen resultados similares a los obtenidos con la elección de Java como base para el lenguaje de ParallelJ. Los usuarios que conozcan el entorno de desarrollo podrán adaptarse sin ningún problema a la utilización de ParallelJ y aquellos que no estén familiarizados con él obtendrán unos conocimientos que probablemente le sean muy útiles posteriormente.

Por estos motivos la herramienta ParallelJ se ha concebido como un pluggin para el entorno de desarrollo Eclipse. Este pluggin extiende las funciones del IDE y lo convierte en un entorno de desarrollo para ParallelJ. A continuación se describe brevemente las características incorporadas a Eclipse.

- Se ha implementado un asistente para la creación de proyectos ParallelJ de forma que el entorno guía al usuario a la hora de crear un nuevo proyecto ParallelJ dentro de la carpeta de trabajo.
- Se proporciona también otro asistente para facilitar la creación de nuevas clases y procesos dentro de los proyectos.
- Para la edición del código fuente, se ha añadido a Eclipse un editor especifico, que no solo cumple las funciones habituales de los editores tales como el resaltado de la sintaxis, el emparejamiento de paréntesis y llaves, la sangría automática,… sino que además proporciona características más avanzadas

como el marcado de errores sintácticos y semánticos muy detallados en el mismo código fuente y mientras se está editando el mismo, sin necesidad de realizar un proceso de compilación o el autocompletado de código para agilizar la escritura del mismo.
- Se ha añadido una página de propiedades a los proyectos que permite configurar todas las opciones de un proyecto ParallelJ como pueden ser la topología de la red o el tiempo que tarda un mensaje en atravesar un enlace.
- Se ha implementado una configuración de lanzamiento que permite ejecutar y depurar proyectos ParallelJ desde Eclipse.
- Se proporciona además un modelo de depuración completo que permite establecer puntos de ruptura en el código fuente desde el mismo editor, lanzar a depuración proyectos y poder inspeccionar todos los procesadores de la arquitectura, los procesos que ejecutan cada uno de ellos y el contenido de las variables de estos y poder avanzar en la ejecución paso a paso.

Todos estos elementos no solo convierten a Eclipse en un entorno de desarrollo para ParallelJ, facilitando todo el proceso implementación y prueba de algoritmos paralelos desde una única herramienta común que permite realizar todas las tareas, sino que también proporcionan al sistema un diseño coherente en todos sus aspectos y una apariencia y funcionalidad añadida realmente avanzadas.

3.5. Licencias

Aparte de todos los apartados ya mencionados ParallelJ presenta una ventaja adicional frente a otras herramientas de similares características. ParallelJ se distribuye actualmente bajo una licencia GPL, en concreto la licencia GPLv3 [2]. La utilización de esta licencia para la distribución de ParallelJ no solo representa que tanto la herramienta como el código se encuentran disponibles para cualquier usuario que las necesiten, sino que además el código de la misma puede ser estudiado y modificado para que quien lo necesite pueda añadir alguna funcionalidad especifica.

Para facilitar la instalación y el uso de la herramienta se encuentra disponible públicamente el manual de usuario de la misma con información detallada sobre el proceso de instalación y puesta en marcha del sistema.

Tanto la herramienta como el manual de usuario se encuentran disponibles en la página web oficial de ParallelJ [8]. En esta misma página aparecerá toda la nueva información que surja alrededor de la herramienta, así como las nuevas versiones de la misma que se vayan desarrollando.

4. Experiencia docente

De poco sirve mencionar las bondades de ParallelJ para la docencia de algoritmia paralela si estas no se contrastan con experiencias reales de utilización de la herramienta por parte de alumnos de este tipo de materias.

Hasta el momento ParallelJ se ha utilizado como herramienta única para la realización de las prácticas de dos asignaturas. La primera experiencia se ha realizado en la asignatura “Algoritmos paralelos”, asignatura optativa de segundo ciclo de Ingeniería en Informática. Gracias al éxito cosechado durante esta primera experiencia se ha extendido el uso de ParallelJ a la asignatura “Programación paralela y distribuida” del Master en computación GRID y paralelismo.

Durante el desarrollo de ambas asignaturas los alumnos han tenido que diseñar, implementar y probar diversos algoritmos utilizando ParallelJ, entre ellos algunos de los más comunes en la materia, como la estrategia maestro-esclavo (n-reinas y camino más corto en grafos), estrategias de paralelismo síncrono mediante barreras locales, lineales y logarítmicas (relajación Jacobiana) o estrategias de pipeline y phased array communication (resolución de sistemas de ecuciones lineales y ordenación bitónica). Todos los problemas planteados a los alumnos han podido ser resueltos utilizando la herramienta ParallelJ.

La adaptación de los usuarios a ParallelJ ha sido más rápida y sencilla que la adaptación a otras herramientas similares que se han utilizado anteriormente en estas asignaturas. Esto es debido a que gran parte de los alumnos disponían de conocimientos previos del lenguaje de programación Java y el entorno de desarrollo integrado Eclipse, al contrario de lo que sucede al utilizar herramientas como Multipascal. Prueba de ello es que mientras que en cursos anteriores la realización de las prácticas debía relegarse tres

sesiones en las que se explicaba al alumno la herramienta Multipascal y las bases del lenguaje de programación Pascal con el uso de ParallelJ ha bastado una sola sesión para explicar las particularidades del pluggin de Eclipse y de las nuevas instrucciones proporcionadas por el lenguaje.

Los conocimientos adquiridos por los alumnos han sido superiores a los obtenidos por los alumnos de cursos anteriores. Todo esto se ha puesto de manifiesto en los informes de prácticas realizados por los alumnos. Esto ha sido posible en gran medida gracias a que los alumnos tenían a su alcance la posibilidad de realizar pruebas más masivas y variadas que con otros sistemas y obtener unos resultados más fáciles de interpretar, lo que ayuda a la comprensión de los conceptos y técnicas de la algoritmia paralela.

En general la transición del sistema Multipascal a la utilización de ParallelJ ha resultado totalmente positiva. Cabría finalmente destacar la satisfacción mostrada por el alumno en las encuestas realizadas con respecto a las herramientas docentes puestas a su disposición (ParallelJ) en contraposición al continuo descontento que se mostraba anteriormente en las asignaturas con respecto a Multipascal.

5. Conclusión

Se ha presentado el recurso docente ParallelJ y sus principales características comparándolas con otros recursos disponibles para la misma materia.

Para aquellas personas que se encuentren interesadas en la herramienta es recomendable permanecer atentos a la página oficial puesto que se continúa trabajando en mejorar el sistema. En futuras versiones se espera incluir algunas características importantes como la posibilidad de importar y exportar proyectos ParallelJ desde Eclipse o mejorar el comportamiento de la red para que las simulaciones se acerquen aún más al comportamiento real de un sistema paralelo de las características escogidas, puesto que la versión actual presenta algunas características poco realistas a la hora de simular la congestión que se produce en la red, debida al tráfico de mensajes producido por la ejecución de los algoritmos diseñados por el usuario.

Si se desea utilizar ParallelJ para la docencia de alguna materia relacionada con la algoritmia paralela tanto la herramienta como el manual de usuario se encuentran disponibles en la página web de la misma. Cualquier duda que pueda surgir sobre la misma o sugerencia sobre posibles mejoras para futuras versiones pueden enviarse a la dirección de correo electrónico que aparece en la misma.

Agradecimientos

ParallelJ no podría existir sin el trabajo y la ayuda de muchas personas. En especial de María Frades Castro como parte fundamental del equipo de desarrollo. El diseño de todas las imágenes que se utilizan en ParallelJ, tanto del logotipo de la herramienta como de los iconos que aparecen en la misma, y la página web oficial de la misma son obra de José Manuel García Marcos. Gracia a la colaboración de los alumnos de la asignatura “Algoritmos Paralelos” la herramienta contiene ahora menos errores y su utilización para otras materias es mucho más factible.

Referencias

[1] Eclipse. http://www.eclipse.org/
[2] GPLv3. http://www.gnu.org/licenses/gpl.html
[3] Hansen, Per Brinch. Superpascal: a publication language for parallel scientific computing. http://brinch-hansen.net/papers/1994c.pdf 1994
[4] Java. http://java.sun.com/
[5] JFreeChart. http://www.jfree.org/jfreechart/
[6] Lester, Bruce P. The art of parallel programming. P-H 1993
[7] MPI. http://www-unix.mcs.anl.gov/mpi/
[8] ParallelJ. http://sfc.unex.es/ParallelJ
[9] PVM. http://www.csm.ornl.gov/pvm/
[10] RMI. http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp