Actas de las I Jornadas sobre Innovación Docente y Adaptación al EEES
en las Titulaciones Técnicas, Granada 2010, pp. 187-190
ISBN 978-84-92757-64-0 DL GR 3336-2010
Ed. Godel Impresores Digitales S.L.

PARALELIZACIÓN DE ALGORITMOS EVOLUTIVOS COMO PRÁCTICA PARA UNA ASIGNATURA DE ARQUITECTURA DE COMPUTADORES

MORA, Antonio Miguel; GARCÍA-ARENAS, Maria Isabel; CASTILLO, Pedro; MERELO, Juan Julián; JIMÉNEZ LAREDO, Juan Luís; GARCÍA SÁNCHEZ, Pablo

Departamento de Arquitectura y Tecnología de Computadores. ETSIIT, Universidad de Granada.
{amorag, maribel, pedro, jmerelo, juanlu, pgarcia}@geneura.ugr.es

Resumen

Este trabajo presenta una propuesta para la consecución de la parte práctica de la asignatura Arquitecturas Paralelas de Altas Prestaciones, una de las nuevas asignaturas del grado en informática dentro de la Universidad de Granada, la cual estará incluida dentro del tercer curso de la especialidad de Ingeniería de Computadores.
En ella, se propone que los alumnos construyan un multicomputador que incluya varios ordenadores independientes unidos mediante una red local. Para ello, dispondrán una serie de puertos de comunicaciones y ejecutarán un algoritmo evolutivo paralelo. Posteriormente analizarán los resultados obtenidos, sobre todo centrándose en la parte concerniente a la ganancia de velocidad de ejecución conseguida, aunque considerando también la mejora en la calidad de las soluciones encontradas. De manera que los alumnos podrán comprobar lo que ocurre realmente al paralelizar algoritmos en sistemas reales, en lugar de utilizar los ampliamente conocidos simuladores.

Palabras clave

Arquitectura de Computadores, Multicomputadores, Algoritmos Evolutivos Paralelos.

1. INTRODUCCIÓN

Por lo general, los alumnos de informática suelen realizar las prácticas de sus asignaturas considerando un nivel de abstracción que los aleja de la visión real que se pretende que aprendan.
Por ejemplo, dentro del área de Arquitectura de Computadores, son de uso común los simuladores de procesadores, tales como WinDLX, DLXV o SuperDLX, los cuales permiten simular arquitecturas DLX [Bestavros-95] ya sean estándar, vectoriales o superescalares.
En la misma tesitura nos encontramos a la hora de que los alumnos trabajen con multiprocesadores o multicomputadores, ya igualmente que se tienen simuladores como ISIS o Augmint ISIS para la arquitectura del primer caso, y Talismán o PP-MESS-SIM para la del segundo. Todos ofrecen la simulación de una red de computadores que es capaz de ejecutar un programa paralelo. Incluso permiten cambiar características de la arquitectura que simulan, como el tipo de red que une los computadores o la potencia y propiedades de cada uno de ellos.
La escasez de medios y las restricciones que se deben imponer a los alumnos a la hora de manejar las aulas, así como los equipos de los que se dispone, hacen difícil plantear prácticas en las que puedan realmente construir un multiprocesador o un multicomputador ellos mismos.
Por eso, y como propuesta dentro de la nueva asignatura para el grado de informática *Arquitecturas Paralelas de Altas Prestaciones*, se ha diseñado una guía para una práctica que será abordada como un trabajo autónomo de los alumnos. En dicha práctica los alumnos podrían crear su propia arquitectura paralela, concretamente un

multicomputador, y podrían ver qué ocurre conforme se van añadiendo nodos de ejecución a dicha arquitectura. De esta forma comprobarían por ellos mismos el efecto de un escalado real.

2. PROPUESTA DE PRÁCTICA

Para ilustrar la construcción de un multicomputador, se propone una práctica en la que, utilizando los computadores de un aula de prácticas (dentro de la Universidad de Granada), un alumno debe construirse un multicomputador con, al menos, 4 computadores/nodos y realizar mediciones del rendimiento de dicho multicomputador al ir aumentando el número de ellos. Para medir el rendimiento de este multicomputador se les propone a los alumnos que utilicen un algoritmo evolutivo paralelo [Eiben-03,Cantú-Paz-00], que se les proporciona ya programado y en el que ellos simplemente tienen que realizar la parte relativa al reparto de las islas en los nodos de los que disponen, es decir, deben distribuir los individuos con la posible solución a un problema, agrupándolos en cada nodo de computación, a fin de que vayan mejorando (aproximándose a la solución que se busca), estableciendo durante la búsqueda un mecanismo de comunicación entre dichas islas.

Igualmente deberán establecer algunos parámetros del algoritmo, los cuales se dejan en principio abiertos, con el fin de que sean ellos mismos los que investiguen el efecto de cada uno de dichos parámetros en la ejecución del experimento.

El esquema del multicomputador que deben construir los alumnos se muestra en la siguiente figura.

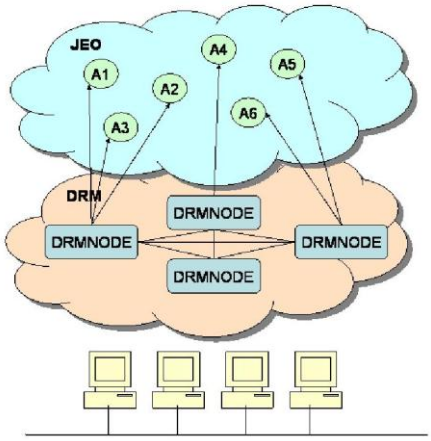

Figura 1. Esquema general por capas del multicomputador que los alumnos deben construir

En dicha Figura se pueden apreciar tres partes bien diferenciadas: la parte más baja, en la que se sitúa la *red real de computadoras* que serán un máximo de 4 nodos unidos por una red local (con las comunicaciones entre los diferentes computadores). En la parte central se sitúa el software instalado en el aula de prácticas que facilita la construcción de una *red de recubrimiento* sobre la red física. Y por último, la parte superior de la figura representa el *software que van a utilizar para la experimentación (algoritmo evolutivo)*, y que deben ejecutar en la máquina paralela que han "construido".

Todo este esquema está basado en las posibilidades que ofrece el proyecto DREAM, ya que se utiliza la máquina virtual que este proyecto presenta, denominada DRM [Jelasity-02] y la librería de algoritmos evolutivos JEO [García Arenas-02].

Los conocimientos que los alumnos poseen sobre computación paralela se afianzarían en esta práctica, que sería un reto para ellos dado que:

- Tendrían que construir un multicomputador estableciendo una red de recubrimiento sobre una red local ya existente que les permita direccionar y controlar los trabajos que van a ejecutar.
- Tendrían que encontrar los parámetros más adecuados para el experimento que quieren ejecutar, realizando pruebas previas y calcular así si con el tiempo de que disponen en las sesiones de prácticas, les será posible recoger los resultados que necesitan para realizar su análisis.
- Se debería encontrar una aproximación/configuración que permita obtener unos tiempos de ejecución mejores con una solución aceptable al problema.

La práctica se organiza en cuatro partes:

- *Construcción del multicomputador*: ejecutarán el programa que nos permite escuchar las comunicaciones de otras máquinas, estableciendo unos puertos de comunicaciones (sockets) en cada nodo del supercomputador (comprobando que dichos puertos no estén ocupados por otro grupo).
- *Prueba de parámetros*: a partir del algoritmo evolutivo que se les ofrece, deberán decidir, para la máquina que han construido, qué parámetros utilizar para la ejecución de los experimentos (tamaño de las islas, tamaño de la población, probabilidades, etc), siguiendo un método de experimentación sistemática, a fin de conseguir soluciones aceptables para el problema que se quiere resolver.
- *Lanzamiento de experimentos*: una vez establecidos los parámetros, realizarán los experimentos con un número de nodos de ejecución variable, que irá desde 1 al número máximo de máquinas que hayan incluido en el supercomputador. Recogiendo los resultados obtenidos en estos experimentos para su análisis.
- *Análisis de resultados*: en base a dichos resultados, evaluarán la máquina paralela que han construido, indicando si es escalable, y si la solución al problema que se ha encontrado es lo suficientemente buena o podría ser mejorada ajustando alguno de los parámetros considerados o que no hayan tenido en cuenta.

3. MATERIAL DOSPONIBLE Y OBJETIVOS

Aparte de una guía de desarrollo de la práctica y del hardware disponible en el aula de prácticas en la que se desarrollará, se deberán cumplir una serie de requisitos en cuanto al software instalado en dichas máquinas a fin de poder realizar esta práctica.

Por lo que será obligatorio tener instalado el *JDK (ver. 6)* del lenguaje de programación Java. Se aconseja también el tener disponible algún entorno de desarrollo como por ejemplo *NetBeans*, en cada nodo de computación debería estar disponible el programa/demonio *drmd* (parte del paquete del proyecto DREAM), con el que conseguirán sincronizar los equipos y pasar mensajes a través de los puertos (sockets). Por último, es necesario disponer de los ficheros de la librería de evolutivos *JEO* [García Arenas-02], con los que podrán probar los algoritmos sobre el multicomputador.

Los objetivos de la práctica pasan por crear una automatización de la definición del multicomputador (mediante un script o similar), así como analizar y comentar todos los aspectos de la consecución de la práctica; desde la creación del multicomputador y su arquitectura, hasta la configuración de los algoritmos (justificando los valores de los parámetros) y el análisis de los resultados obtenidos.

4. RESULTADOS OBTENIDOS Y CONCLUSIONES

La práctica ha sido probada por dos grupos de profesores (de 4 y 3 personas), creando cada grupo un multicomputador con 4 nodos cada uno. Para ello utilizaron sólo una parte del aula de prácticas, concretamente 6 máquinas, para así utilizar al menos 2 nodos solapados en los supercomputadores.

Dentro de la parametrización se eligierón cuántas poblaciones/islas (procesos) lanzar en el multicomputador, a fin de evaluar mejor las prestaciones, así como el número de individuos por población, y el número de generaciones que deben evolucionar cada proceso hasta encontrar una solución aceptable.
Algunos de los resultados obtenidos se muestran en las Figura 2.

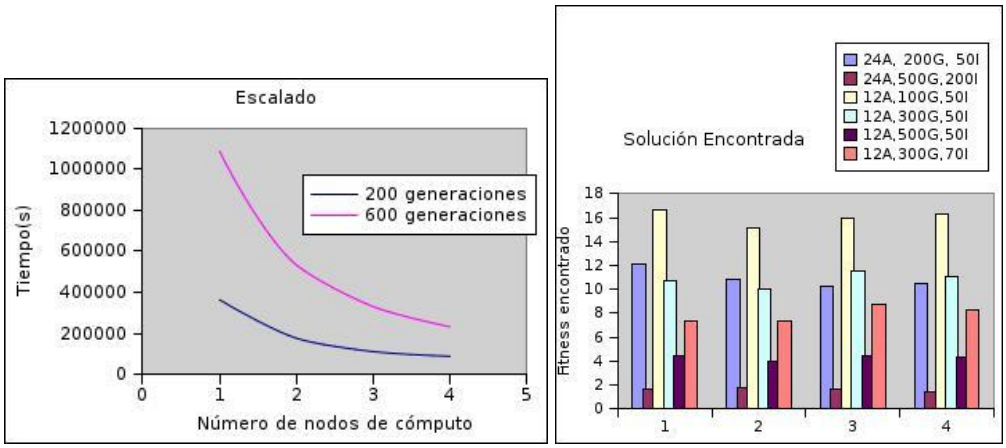

Figura 2. (Izquierda) Resultados de escalabilidad para 12 poblaciones de 50 individuos. (Derecha) Soluciones encontradas para el problema (considerando 12 y 24 poblaciones).

En ella se puede ver que la reducción de tiempos obtenida en ambos casos (para distinto número de generaciones) es quasi-exponencial en los tiempos conforme aumenta el número de nodos de cómputo. En cuanto a las soluciones encontradas para el problema a resolver (la optimización de la función de Ackley [Ackley-85], cuya mejor solución es 0), se puede ver que el número de poblaciones es poco determinante en este caso, siendo el factor más importante el número de generaciones, siendo 500 la que mejores resultados ofrece (y más tiempo emplea, por supuesto).
Como conclusión final, comentar que vistas las posibilidades de creación y configuración de un multicomputador que se ofrecen al alumno, mediante herramientas y métodos usados actualmente dentro del ámbito de la investigación, creemos que el desarrollo de estas prácticas motivaría a los estudiantes y promovería el espíritu de autoestudio hacia el que se debe orientar la docencia en los nuevos títulos de grado.

Bibliografía

Bestavros A., Liu Y. (1995). "Simulation of hardware dynamic scheduling on the DLX Architecture", Tech. Rep., ACM, Digital Library.

Eiben A.E., Smith J.E. (2003) "Introduction to Evolutionary Computing", SpringerVerlag

Cantú-Paz. E. (2000). "Efficient and Accurate Parallel Genetic Algorithms". Kluwer Academic Publishers.

Jelasity M., Preub M., Van Steen M., Paechter B. (2002). "Maintaining connectivity in a scalable and robust distributed environment", 2nd IEEE International Symposium on Cluster Computing and the Grid (CCGrid2002), p. 389-394.

García Arenas M.I, Foucart L., Shoenauer M., Merelo J.J. (2002). "Computación evolutiva en Java: JEO", Actas de AEB 2002, Ed. Universidad de Mérida, p. 46-53.

Ackley D.H. (1985). "A connectionist algorithm for genetic search", en Proc.of Inter. Conference on Genetic Algorithms, Ed. J.J. Greffenstette, p. 121-135.