Reparto de carga en servidores para Internet

Martín Moreno, Ismael; Rodrigo Yanes, Juan Antonio
Grupo de Sistemas Distribuidos y Aplicaciones Telemáticas, jrodrigo@aut.uah.es, telf. 34 91 885 65 03

Gutiérrez de Mesa, José Antonio
Depto. Ciencias de la Computación, jgutierrez@uah.es, telf. 34 91 885 66 49

Arribas Aranda, Jordán; Fernández Rodríguez; María del Carmen
Grupo de Sistemas Distribuidos y Aplicaciones Telemáticas

Resumen

En este trabajo se presentan los resultados obtenidos en la realización de un robot para búsqueda de páginas y ordenación de la información en Internet[1]. La arquitectura realizada es multihilo. El estudio de las páginas se realiza teniendo en cuenta que actualmente la mayoría de las páginas web existentes son interpretadas partiendo de un texto plano estático o dinámico, según algunas de las nuevas tecnologías (JSP, servlets, ASP, ...). También se ha desarrollado un sistema de caché para permitir una mayor fluidez en el acceso a los datos y, por tanto, una mayor eficacia. El desarrollo se ha realizado en lenguaje JAVA. El módulo de almacenamiento de los datos está soportado por la base de datos MySQL

1. Introducción

Cuando se habla de Internet, se suele describir en primer lugar la gran cantidad de páginas con información disponible que hay en ella, pero la parte más complicada (las búsquedas) queda siempre en un segundo plano, como desagradable sorpresa para quienes se conectan por primera vez. Es el navegante el que tiene que ingeniárselas para dar con la información deseada. En muchas ocasiones, la información y la búsqueda en sí están llenas de paradojas y situaciones extrañas, muchas veces frustrantes.

Lo que verdaderamente ha cambiado con la llegada de Internet es que toda la información de la Red existe en formato digital y que miles de ordenadores de todo el mundo la almacenan públicamente. Gracias a ello, se puede buscar cualquier palabra o concepto y acceder a ella en cuestión de segundos. Lo mejor de todo es que no hace falta ningún programa especial, existen "buscadores de Internet" a los que acceder para localizar la información.

Con la llegada de nuevas tecnologías y de mejores conexiones, también aparecieron nuevos sistemas más potentes que recopilaban toda la información de la World Wide Web y de Usenet: los motores de búsqueda [2]. Encabezados por AltaVista, de Digital, los servicios de este tipo emplean "robots" inteligentes que saltan de una página a otra de la Web (a través de los enlaces de hipertexto) recogiendo páginas y almacenando toda la información en una gigantesca base de datos.

El éxito de Altavista se basó en un potente robot que inspecciona más de tres millones de páginas diarias y un enorme sistema de indexación que almacena las páginas.

Otros sistemas de búsqueda famosos son[3]:

Lycos (http://www.lycos.com), diseñado en la Universidad Carnegie Mellon. Lycos ordena por título, por la cabecera

Este trabajo ha sido financiado como parte del Proyecto de Investigación CICYT Ter-98-0544.

del documento, las cabeceras y subcabeceras, los enlaces, las100 palabras más utilizadas del documento y las primeras 20 líneas.

Yahoo! (http://www.yahoo.com), Está dividido en áreas temáticas que a su vez se subdividen en otras jerárquicamente inferiores. Se actualiza diariamente y ofrece diferentes servicios: la lista de webs añadidos durante la semana anterior; una selección de los webs más interesantes donde se puede encontrar de todo; una recopilación de noticias de actualidad que se renuevan cada hora, etc.

Estos sistemas suponen una evolución de la solución adoptada inicialmente por Altavista, que presenta la lista de las referencias que cumplen el criterio de búsqueda sin ninguna organización, hacia una presentación en la que la lista de referencias se organiza en forma de árbol. Bien es cierto que, de momento, el árbol se utiliza para reducir el campo de búsqueda, es decir, como una forma de ayuda a los usuarios para enfocar la consulta.

La aparición de XML y su uso, se espera que sea de gran ayuda para el trabajo de los motores de búsqueda, al facilitar y acelerar el proceso de búsqueda de los elementos claves del documento para su indexación[4]. XML obliga a definir y cumplimentar campos de identificación del documento[5]. Pero todavía falta mucho trabajo de estandarización de estos campos.

Otra línea de evolución de los motores de búsqueda está en la adaptación de su trabajo a las necesidades del usuario, bien suministrando información no directamente relacionada con las búsquedas concretas del usuario, bien concentrando las búsquedas en el perfil del usuario. Este perfil se debe construir en base a las búsquedas que realiza el usuario [6].

2. Motores de búsqueda

Un motor de búsqueda, o lo que es lo mismo, un robot, es un programa residente en un ordenador conectado a Internet que recupera de forma automática los títulos, las cabeceras y/o el texto de las páginas web. A partir de estos

elementos genera unos índices de palabras clave que se pueden buscar desde los clientes WWW. Es decir, su tarea principal consiste en crear un listado de direcciones URL en una base de datos para poder ser consultadas en un momento dado.

El módulo que realiza estas peticiones se conoce como "buscador web". El buscador web no se encarga de recopilar los enlaces a través de la web, sino de acceder a la base de datos mediante una consulta para recuperar los datos solicitados. Por tanto un motor de búsqueda está formado por cuatro partes principales:

- Un robot que "recorre" Internet buscando información y las referencias a la misma.
- Un sistema automático de análisis de los documentos y de ordenación de sus contenidos.
- Una interfase de consulta, generalmente basada en el lenguaje de consulta de bases de datos SQL.
- Un módulo de enlace entre la base de datos y el generador de consultas.

3. Java, hilos y bases de datos

El lenguaje de utilizado en el desarrollo de este trabajo ha sido Java[6]. La razones principales para esta elección fueron: la facilidad de diseño de aplicaciones multihilo y de aplicaciones para redes de ordenadores y la portabilidad de los ejecutables. La velocidad de ejecución, la prinicpal desventaja, no se primó en esta fase del trabajo en la que los resultados, al comparar soluciones, son relativos.

Java es un lenguaje orientado a objetos, interpretado, pero no distribuido. Su adecuación al diseño de aplicaciones distribuidas le viene de las librerías, librerías que facilitan las conexiones TCP/IP y el uso de protocolos como HTTP y FTP para la construcción de aplicaciones que intercambien información a través de una red.

Las posibilidades de trabajo en máquinas no uniformes, siempre que se disponga de la máquina virtual para todos los sistemas interconectados, facilita el diseño, al permitir que el mismo código pueda trabajar en máquinas no homogéneas.

El beneficio de ser multihilo consiste en un mejor rendimiento interactivo y en un mejor comportamiento en tiempo real. Aunque el comportamiento en tiempo real está limitado a las capacidades del sistema operativo subyacente (Unix, Windows, etc.), que aun supera a los entornos monoprogramados (un único hilo) tanto en facilidad de desarrollo como en rendimiento.

Otra ventaja de Java respecto del desarrollo completo es que no intenta conectar todos los módulos que comprenden una aplicación hasta el tiempo de ejecución. Esto facilita la incorporación, actualización, de las librerías nuevas, ya que no hay que modificar las aplicaciones actuales (siempre que mantengan el API anterior).

Para relacionar las aplicaciones con las bases de datos, Java dispone de JDBC. Este tipo de interfase es genérica, y la proporciona el desarrollador de la máquina virtual (Sun, Microsoft, Netscape,....). JDBC utiliza los controladores ODBC como medio de acceso a la Base de Datos. Funciona aplicando el API JDBC sobre el API ODBC.

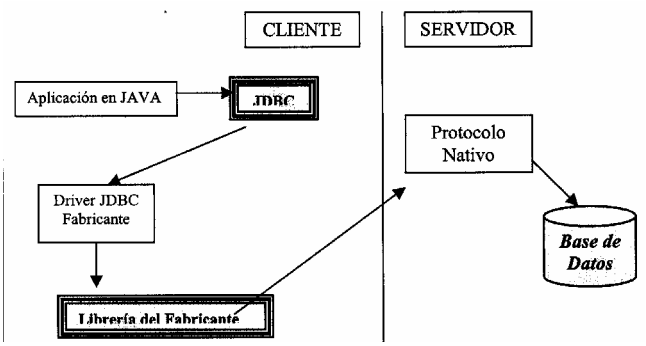

Fig. 1. *Arquitectura de las aplicaciones con Java y JDBC.*

4. Análisis del sistema

Dado que el robot se diseña con fines de investigación, debe cumplir ciertas condiciones, que no son tan necesarias en un robot comercial, para asegurar su control y analizar su rendimiento. El robot debe ser capaz de analizar una serie de páginas web hasta un nivel predeterminado por el usuario. Para acceder a estas páginas el motor deberá ser capaz de conectar con un servidor determinado y, mediante peticiones, obtener los recursos solicitados (páginas HTML) que serán analizados para encontrar coincidencias de las palabras del diccionario y nuevos enlaces a otras páginas [8]. También deberá medir una serie de tiempos que permitan estudiar cada uno de los módulos independientemente. Los datos resultantes de este análisis serán almacenados en unas tablas de la base de datos donde, posteriormente, serán analizados.

Con este fin, al robot se le ha incorporado una interfaz gráfica donde el usuario puede configurar todos los parámetros precisos para el funcionamiento del motor [9].

La arquitectura de ejecución está basada en hilos idénticos que ejecutarán la misma tarea sobre distintos datos. Así conseguimos realizar varias operaciones a la vez sin que exista interferencia entre ellas.

Esta arquitectura puede desarrollarse de dos formas diferentes:

- Una serie de hilos "idénticos" "recorrerán" Internet buscando los enlaces y las coincidencias de las palabras seleccionadas. Cada uno de estos hilos, a su vez, generará tantos hilos como enlaces distintos haya encontrado en el texto HTML, siguiendo así una búsqueda en árbol.
- Un proceso principal gestionará toda la carga de hilos y el control del flujo del programa. Este proceso principal tendrá la misma duración que la ejecución del motor y durante todo este período será el encargado de ejecutar las tareas que realizan la búsqueda.

La segunda opción ofreció mejores resultados y se optó por ella. Los inconvenientes detectados en la primera son:

- Dificultad de tener controlados todos los hilos del sistema. Al no existir un registro general de los hilos ejecutándose en el sistema en un momento determinado, era difícil detectar si uno de ellos se bloqueaba indefinidamente.

- Lanzamiento masivo de hilos. En un período pequeño de tiempo se puede lanzar un gran número de hilos, lo cual produce una sobrecarga de tareas en el sistema que disminuye el rendimiento

Los módulos de que consta la arquitectura actual son:

- Descarga de páginas.
- Búsqueda de enlaces y de palabras.
- Control de estado e hilos.
- Almacenamiento en la base de datos.
- Control de la caché.
- Interfaz gráfica.

El módulo de control de estado e hilos es el núcleo de la aplicación. Él se encargará de controlar en todo momento el estado del sistema y el desarrollo de la búsqueda. También se encargará de mostrar los datos relativos al proceso y de controlar el lanzamiento de los distintos hilos según se vayan necesitando. En este módulo se establecerá el inicio y fin del motor y de él partirá toda la carga al sistema.

Para optimizar el tiempo de acceso a los datos se ha introducido un módulo de caché. Este módulo se encarga de almacenar en ficheros locales las páginas que se vayan analizando para que si posteriormente son solicitadas, sea más rápida la descarga

5. Resultados y ajuste

Como en toda aplicación, las primeras pruebas estuvieron encaminadas a ajustar los parámetros que controlan la ejecución del robot [10].

Estos parámetros son los siguientes:

- Tiempos medios de los módulos: Es necesario conocer los recursos que la aplicación necesita y determinar en qué emplea más tiempo cada hilo durante el procesamiento de un enlace. Para ello se ha realizado la prueba lanzando el motor sobre distintas direcciones base y se han calculado los tiempos medios de cada uno de los bloques principales de la aplicación.

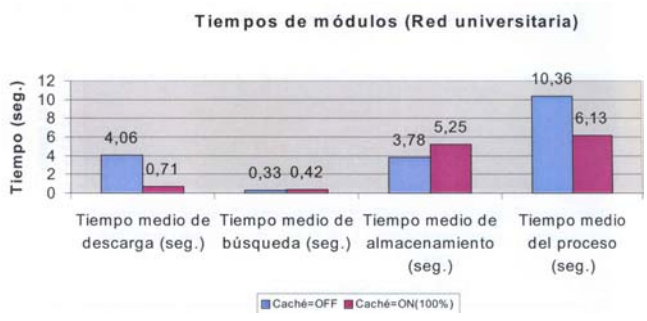

Fig. 2. *Tiempos de los módulos en la red de la universidad.*

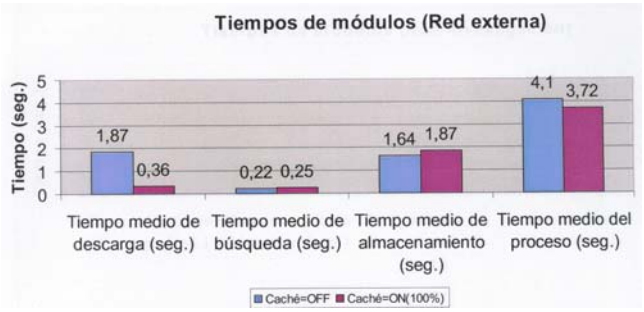

Fig. 3. *Tiempos de los módulos en Internet.*

- Número de hilos en ejecución: El objetivo de esta medida es encontrar el valor adecuado del número máximo de hilos que pueden ser lanzados concurrentemente sin que aparezcan problemas por exceso de trabajo en la CPU, es decir, evitar que un valor elevado impida el lanzamiento de nuevos hilos, y que un valor excesivamente bajo provoque una acaparación de recursos por parte del proceso principal.

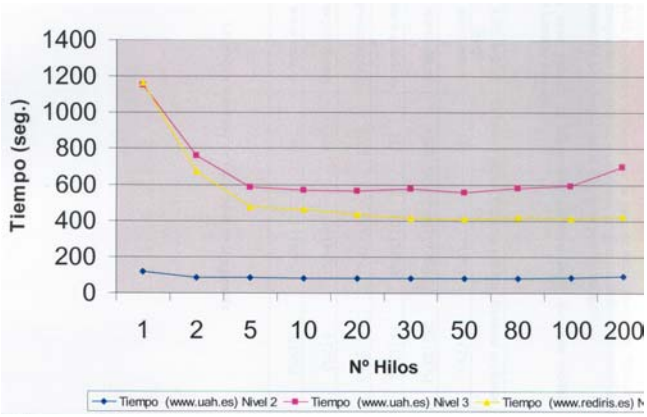

Fig. 4. *Influencia del número de hilos sobre el tiempo de ejecución.*

Con esta prueba también se estudió el valor más adecuado para el tiempo entre lanzamientos de hilos. Es necesario estudiar este tiempo porque no conviene lanzar varios hilos al mismo tiempo, ya que se puede disminuir sensiblemente el rendimiento del sistema, pues todos los hilos lanzados a la vez competirán por el mismo tipo de recursos (CPU, descarga de ficheros, acceso a la base de datos, etc.) en el mismo momento.

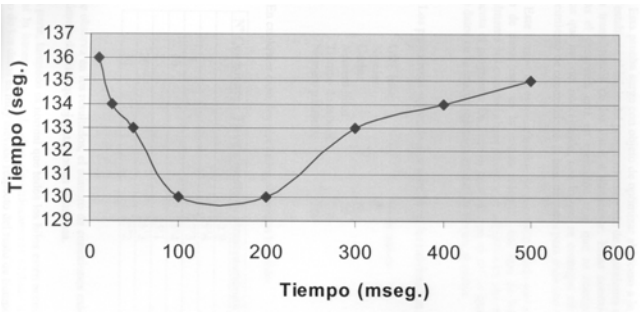

Fig. 5. *Influencia del tiempo de espera entre generación de hilos y el tiempo de ejecución.*

Las pruebas se realizaron con los siguientes valores: URL base: www.uah.es, niveles: 2, caché: no, número hilos: 10, conexiones con la BD: 10, número de palabras: 100.

De los resultados se deduce que la influencia del tiempo entre lanzamiento de hilos no es significativa si nos mantenemos cerca del valor óptimo, que se sitúa alrededor de los 130 milisegundos.

- El siguiente ajuste trata de optimizar el acceso a la base de datos. Para ello se establecen al inicio del programa una serie de conexiones con la base de datos para la transferencia de los datos a almacenar. Estas conexiones no se cierran mientras el programa está en ejecución, sino que se reasignan a los distintos procesos que las van necesitando, con lo que se consigue eliminar el tiempo de establecimiento de conexión con la base de datos para cada uno de los hilos.

Este tiempo de establecimiento de conexión puede variar dependiendo de si la base de datos está en la misma máquina, del tipo de base datos, etc. Es especialmente útil cuando la base de datos se encuentra en una máquina distinta a la que ejecuta el programa.

Las pruebas se realizaron con los siguientes valores: URL base: www.uah.es, niveles: 2, caché: no, número hilos: 20, tiempo de espera: 200, número de palabras: 250.

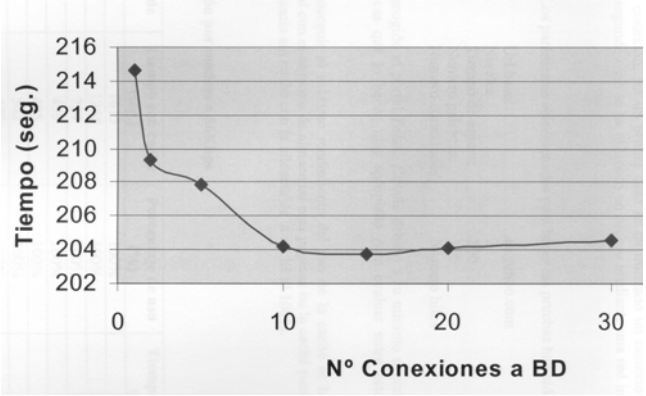

Fig. 6. *Relación entre el número de conexiones abiertas y la velocidad de acceso a la base de datos.*

Como se observa en los resultados, el número de conexiones más adecuado es algo menor que el número de hilos que ejecutamos en el sistema. Por otro lado, no tendría sentido que el número de conexiones fuese superior al de hilos ya que cada hilo sólo utiliza una conexión en cada momento.

- Eficacia de la caché: Se ha desarrollado un sistema de caché que permite leer del disco duro local aquellas páginas a las que ya se ha accedido anteriormente y cuya versión en la web no ha sido actualizada. Este sistema es especialmente útil cuando se dispone de un número de páginas elevado en la caché o la conexión no es especialmente rápida.

Para realizar estas pruebas se han utilizado dos máquinas distintas con distintas velocidades de conexión. En un primer caso se ha utilizado un conexión por módem a 56Kbps., en el segundo caso se ha utilizado un acceso mediante una red interna conectada a fibra óptica.

Para conseguir el máximo rendimiento del uso de la caché se ha realizado una búsqueda inicial con el objetivo de almacenar esas paginas en la caché para posteriormente comparar la eficacia sin caché con la eficacia de la caché al 100%..

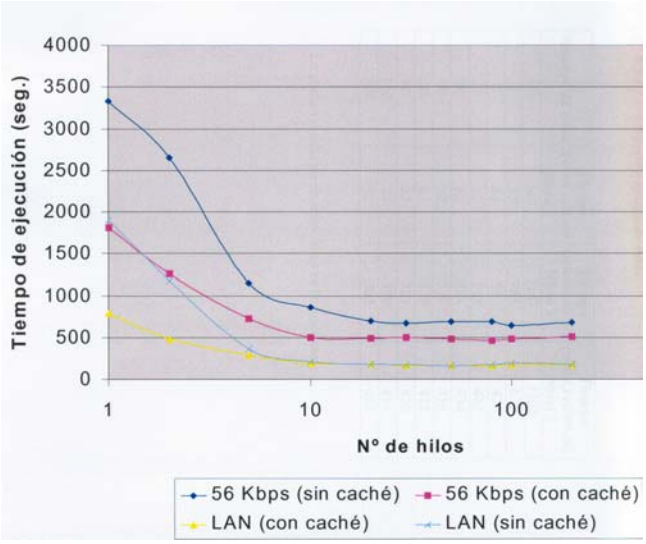

Fig. 7. *Influencia de la caché en el tiempo de ejecución.*

En la gráfica se observa que el rendimiento de la caché es considerablemente mayor en el caso de que la velocidad de conexión sea baja. En cualquier caso, e independientemente de la velocidad de conexión, se puede apreciar que para un número de hilos bajo el rendimiento que ofrece la caché es mucho mejor que para un número de hilos alto. Esto se debe a la capacidad de multitarea del sistema que permite realizar la descarga de varias páginas simultáneamente mientras se procesan otras, lo cual disminuye el factor de aprovechamiento de la caché.

- Número óptimo de hilos: De todos los valores anteriores se deduce que el rendimiento debe crecer con el número de hilos hasta un determinado valor, a partir del cual el rendimiento debe disminuir, ya que los nuevos hilos producirán sobrecarga sin realizar trabajo útil.

En la gráfica (8) se observa que la degradación al aumentar el número de hilos tiene una pendiente muy pequeña. Esto era de prever en máquinas que aceptan el trabajo de un número considerable de hilos sin que se vea afectado el rendimiento global. Obsérvese que la degradación se empieza a notar por encima de los 80 hilos.

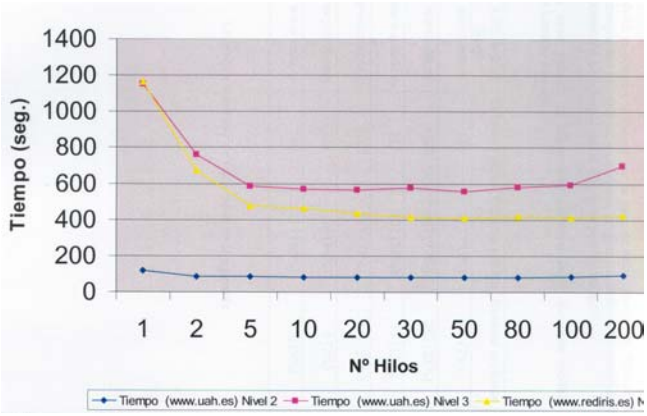

Fig. 8. *Tiempos de ejecución en función del número de hilos.*

- URL válidas frente a URL erróneas: Esta relación mide la eficacia del módulo buscador de enlaces. Se puede comprobar que el grado de efectividad de este módulo

es muy elevado ya que prácticamente el 90% de las páginas accedidas son válidas.

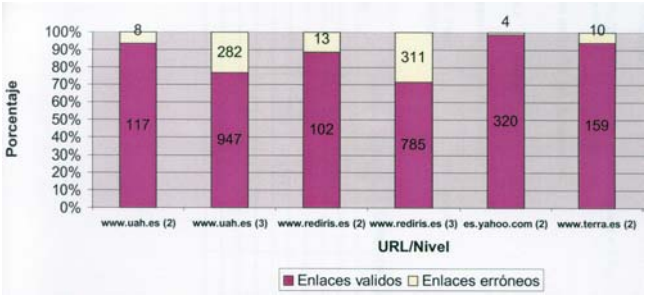

Fig. 9. *Relación entre páginas válidas e incorrectas.*

6. Conclusiones y futuro trabajo

En este trabajo se han presentado los resultados obtenidos con el robot de búsqueda en Internet. El robot se ha analizado, como sistema multiproceso, como un primer paso que concluirá con el diseño de forma multimáquina.

Del robot multimáquina ya se han realizado dos versiones, que se están evaluando.

Del funcionamiento del robot multiproceso podemos destacar:

Uno de los factores más importantes en el funcionamiento de esta aplicación es el ancho de banda de conexión para ejecutar el motor. Como se ha podido comprobar en las pruebas, un porcentaje muy elevado del tiempo de la aplicación se dedica a descargar el código HTML de Internet. Para mejorar este apartado se ha creado el módulo de caché.

Respecto a estos resultados, es importante destacar el elevado rendimiento que se obtiene del módulo de caché cuando el número de hilos es menor de 10, siendo prácticamente inapreciable la mejora de tiempo para el resto de los casos. Esto es debido a las propias características de la multitarea que permiten ejecutar distintas operaciones mientras otras están bloqueadas o esperando datos de una E/S.

Otro de los elementos determinantes a la hora de mejorar la eficacia de la aplicación es el tema del acceso a la base de datos. Para optimizar este apartado se ha creado un "buffer" de conexiones abiertas con la base de datos para evitar que cada hilo deba abrir y cerrar continuamente estas conexiones, con su consecuente pérdida de tiempo en la ejecución. La creación de este módulo también ha quedado justificada tras analizar los resultados obtenidos, ya que una correcta elección del número de conexiones con la base de datos puede mejorar el tiempo de ejecución de la aplicación.

En cuanto a los tiempos obtenidos para cada uno de los módulos se ha observado que casi la totalidad del tiempo que dura la ejecución de un hilo se lo reparten el tiempo de descarga de Internet y el tiempo de almacenamiento de los resultados en la base de datos.

Actualmente, estamos trabajando en la distribución del robot dentro de una red de ordenadores. También se están estudiando los diferentes sistemas de ordenación para permitir el acceso de los usuarios a la información recogida por el robot.

Referencias

[1] I. Martín , J.A. Rodrigo, "Motor de búsqueda en Java", *Univ. de Alcalá, Alcalá de Henares, 2001*

[2] D. Casals, J.A. Rodrigo, *"Robot buscador de información"*, *Univ. de Alcalá, Alcalá de Henares, 2000*

[3] *F. Marckini, "El posicionamiento en buscadores"*, Dany Press, Madrid, 2001

[4] *J.A. Rodrigo, "Motores de búsqueda utilizando XML"*, Apuntes del curso "XML", Alcalá de Henares, 2001

[5] *J.A. Gutiérrez, R. Barchino, J.M. Gutiérrez "Posibilidades del comercio electrónico con agentes virtuales utilizando XML"*, Univ. Alcalá, Alcalá de Henares, 2001

[6] C.A. Iglesias, M. Garijo, J.C. González. *"A survey of Agent-Oriented Methodologies"*, en "Inteligent Agents V", Springer 1555, Berlín, 1998, pag. 317-30.

[7] *J2SDK1.3, Manual de usuario*, SUN Microsistems, 2001

[8] *F. Menczer y A.E. Monge, "Scalable Web Search by Adaptative Online Agents: An InfoSpiders Case Study"*, en "Intelligent Information Agents", Springer, Berlín 1998, pag. 323-47

[9] *E.J. Glover. S. Lawrence, M.D. Gordon, W.P. Birmingham and C.L. Giles "Web Search - Your Way" ACM Communications, vol. 44, nº 12, Diciembre 2001., pag. 97-102.*

[10] *J. Williams. "Bots and other Internet Beasties" Sams-Net. 5ª ed. USA. 1996.*