Agentes móviles para extractos de documentos

Lorena Leal Bando, Dames Vilariño Ayala, Fabiola López y López

Benemérita Universidad Autónoma de Puebla, FCC
14 Sur y Av. San Claudio
lore_lb33@hotmail.com, dames@cs.buap.mx, fabiola.lopez@siu.buap.mx

Abstract. La Recuperación de Información (RI) busca extractos o documentos que brinden una respuesta más exacta a la pregunta formulada. Con el surgimiento de Internet se ha acrecentado la necesidad de utilizar de manera efectiva la información disponible en diferentes servidores remotos. Por lo cual se requiere incorporar mecanismos eficientes de búsqueda de información distribuida espacialmente, esto incrementa el tráfico en la red. Uno de los beneficios de la tecnología de agentes móviles es precisamente, la reducción de costos en la comunicación entre los múltiples host que se pueden visitar. El objetivo de este trabajo es demostrar cómo el uso de agentes móviles agiliza la recuperación de información para lo cual usamos como herramienta a ProActive. Se proponen dos diferentes estrategias de migración.

1 Introducción

La RI es un área de las Ciencias de la Computación de gran importancia por dedicarse al estudio de sistemas y técnicas para asignar índices, buscar y devolver datos valiosos a los usuarios. En los últimos años se ha acrecentado la necesidad de utilizar de manera efectiva la información disponible en una red, como puede ser Internet [9]. Actualmente se estima que diariamente se crean 4500 sitios nuevos y sólo 350 de ellos desaparecen. En países como el nuestro, Internet crece a pasos agigantados, considerando que es una tecnología nueva que hizo su aparición hace menos de 15 años. Este crecimiento se ve reflejado en el número de usuarios, hace 3 años sólo el 3% de la población tenía acceso a Internet, ahora se estima que para el año 2005 este porcentaje se incrementará en un 13%, lo que representa 17 millones de usuarios en México, expresado por Rodolfo Sandoval, Director de Intel en México [1, 9, 13]. Otra razón por la que el tratamiento de la información se vuelve más complejo es debido a la calidad de la misma, ya que en la red se cuenta con diversos tipos de datos, que incluyen texto, imágenes, sonido, etc.

Debido a que la información está espacialmente distribuida (en sitios conectados a Internet), el problema radica en localizar efectivamente los lugares donde se pudiera encontrar información interesante. De ahí la necesidad de incorporar a la RI mecanismos que le permitan acceder a sitios remotos. Una solución inmediata sería el traer los documentos al procesador central para ser analizados. Esto incrementaría enormemente el tráfico en la red porque se tendrían que enviar cualquier documento aún cuando no fuese relevante a la solución.

© A. Gelbukh, H. Calvo (Eds.)
Avances en la ciencia de la computación en México
Research on Computing Science 13, 2005, pp. 63-74

3 Agentes móviles

Un agente es algo que actúa, (agente proviene del Latín *agere* que significa, hacer) [20]. Informalmente un agente es una entidad con objetivos, que alcanza ejecutando una serie de acciones, mediante un dominio de conocimiento y situado en un entorno concreto [6]. Es difícil encontrar una definición exacta para el término "agente" que incluya todas las propiedades y características que los investigadores y diseñadores consideren relevantes, es por ello que existen numerosas definiciones de agentes. A continuación se citan algunas de las más solicitadas dentro de la literatura:

Para Russel y Norving un agente es una entidad que percibe y actúa sobre un entorno a través de efectores [20]. Wooldridge y Jennings: un agente es un sistema informático situado en un entorno y que es capaz de realizar acciones de forma autónoma para conseguir sus objetivos de diseño [23].

Actualmente el paradigma de agentes ha robado la atención de varios investigadores, y su colaboración ha proporcionado gran cantidad de literatura al respecto, incluyendo definiciones, clasificaciones, diseño de arquitecturas, lenguajes y protocolos de comunicación. Existen también organizaciones como la FIPA [22, 26] y la OMG MASIF [22 ,27] que como primer paso trabajan en la estandarización de sistemas de agentes, a su vez de organizar y realizar eventos para congregar a más personas interesadas en el tema. Gracias a estas aportaciones se han desarrollado ya gran cantidad de agentes, que han sido utilizados en control de procesos, comercio electrónico, herramientas de desarrollo, software de interfaces de usuario, manejo de redes y tareas de recuperación de información.

Un agente es caracterizado por una serie de calificativos, los cuales denotan ciertas propiedades a cumplir [20, 23].

- Autonomía, el agente percibe su entorno y puede operar por si sólo sin la intervención de humanos u otros agentes.
- Reactividad, el agente es capaz de responder a cambios en el entorno en el que se encuentra situado.
- Pro-actividad, a su vez el agente debe ser capaz de intentar cumplir sus propios planes u objetivos.
- Sociabilidad, debe de poder comunicarse con otros agentes mediante algún tipo de lenguaje de comunicación de agentes.

En la literatura se suelen atribuir otras características a los agentes en mayor o menor grado para resolver problemas particulares, una de ellas es la movilidad que se define como la capacidad de un agente de trasladarse a través de una red.

Los agentes móviles son procesos computacionales capaces de navegar por redes [4,8,11,12,13,14,15,16,18,22,25] WAN, como Internet, interactuando con equipos, acumulando información en nombre de sus propietarios, y volver después de haber realizado las tareas requeridas por el usuario para informar a este de los resultados. Los atributos de movilidad en los agentes han introducido el concepto de programación remota, donde un mismo agente puede actuar tanto de cliente como de servidor. Este tipo de agentes tienen diversas características que los hacen atractivos los desarrolladores ya que deben ser portables a través de plataformas, capaces de

elegir cuando y donde transportarse a sí mismo, duplicarse, comunicarse con otros agentes para intercambiar información [4,11,12].

Los agentes móviles agregan dos cosas:

- Datos (datos recopilados y estados de procesos) y;
- Código (instrucciones que dirigen la conducta).

Un agente móvil debe ser hábil para ejecutarse en cualquier máquina dentro de una red, independiente del tipo de procesador o sistema operativo. El código del agente no debe haber sido instalado en cada máquina que el agente pudiera visitar potencialmente; este debe moverse con datos del agente [11].

Son ideales por que se reduce la carga de red al evitar el movimiento de datos, se mueven los cálculos hacia los datos y no al revés, se descentraliza los procesos al asignar tareas específicas en diferentes lugares, permiten no estar conectados a la red, mejoran la latencia de red en sistemas de tiempo real y para redes de gran tamaño [12, 14,16].

Por el contrario, tienen una serie de riesgos relativos a la seguridad, la confidencialidad y la autenticación. En primer lugar un agente es representante de un usuario, por consiguiente, se debe garantizar y probar en todo momento la identidad del usuario a quien representa evitando suplantaciones fraudulentas. Además, los agentes no actúan en solitario, sino que se localizan en entornos o plataformas con varios agentes, donde cada uno tiene sus propios objetivos, toma sus decisiones y puede tener la capacidad de comunicarse con otros *agentes*, lo que crea nuevos riesgos en las comunicaciones. Dichos entornos se conocen como sistemas multi-agente [12, 14, 16, 18].

4 ProActive

Para poder desarrollar agentes móviles y que estos puedan migrar entre varios hots, es preciso utilizar plataformas específicas destinadas a tal fin, que permitan a los agentes móviles desplazarse entre las máquinas que componen dicha plataforma. La plataforma debe proporcionar facilidades para el desarrollo del agente móvil, tales como el soporte donde implementar su modelo de ciclo de vida (creación, clonación y migración). Además debe facilitar la comunicación entre agentes (o entre agentes y un usuario u otra entidad) y proporcionar servicios de seguridad para evitar que agentes maliciosos causen estragos.

ProActive es una API que permite desarrollar aplicaciones formadas por varios subsistemas y cada uno de ellos compuesto por Objetos Activos (OA). En ProActive los OA's son unidades básicas de actividad y distribución para aplicaciones concurrentes [28].

El OA está compuesto por dos objetos: un *cuerpo* y un objeto Java estándar. El cuerpo no es visible desde el exterior del objeto activo, y todo luce como si el objeto estándar fuese activo. El cuerpo es responsable de recibir los llamados del OA, los almacena en una cola de peticiones y posteriormente ejecuta estos llamados en un orden específico según la política de sincronización establecida [2, 14, 28].

Para evitar esto se propone el uso de agentes móviles que puedan procesar localmente la información y sólo transmitan aquella que sea relevante [5, 7, 10]. Los agentes móviles son programas que pueden migrar de un host a otro en una red. El objetivo de este trabajo es demostrar cómo el uso de agentes móviles agiliza el tiempo de búsqueda de información, y a la vez permite que sea un proceso menos tedioso para los usuarios. En nuestra propuesta, los agentes migran a varios hosts usando diferentes estrategias de migración, aplican técnicas de RI y regresan sólo con aquella información que sea relevante a la consulta solicitada. Para ello usamos ProActive [28] que es una librería pura de Java que permite realizar aplicaciones paralelas, distribuidas y concurrentes. ProActive nació de un proyecto del Consorcio ObjectWeb y del Instituto Nacional de Investigación en Informática y en Automática (INRIA) en Francia. El artículo está organizado de la siguiente manera. Las secciones 2, 3 y 4 proporcionan los fundamentos teóricos de la extracción de información, conceptos relacionados a los agentes móviles y una descripción general de la plataforma ProActive, respectivamente. La sección 5 da al lector la visión general de la implementación de este trabajo. Por último, en las secciones 6 y 7 se exponen los resultados obtenidos hasta el momento y las conclusiones del mismo, respectivamente.

2 Extractos de documentos

El tesoro más valioso del hombre es la información y sin lugar a dudas el potencial de este tesoro radica en la habilidad para realizar ciertas operaciones con la información. Para procesar una gran cantidad de datos se ha hecho uso de la computadora, pero lo que se tiene que considerar es que no son capaces de entender los textos para su procesamiento, para una computadora un documento sólo representa cadenas de caracteres sin algún sentido. Actualmente los investigadores pretenden hacer de la computadora una herramienta eficiente, sin embargo es necesario dotarla con un aprendizaje de la estructura de textos lo que implica el estudio de ciencias del lenguaje y ciencias de razonamiento [13,14].

El análisis documental está constituido por el conjunto de operaciones necesarias para seleccionar las ideas informativamente relevantes de un documento, a fin de expresar su contenido sin ambigüedades, para recuperar la información en él contenida. Esta representación puede ser utilizada para identificar el documento, para procurar los puntos de acceso en la búsqueda de documentos, para indicar su contenido o para servir de sustituto del documento [13, 14].

Por generación automática de resúmenes de texto se entiende que es el proceso por el cual se identifica información sustancial, proveniente de una fuente o varias para producir una versión abreviada. Un resumen debe ser claro, ordenado, expresa lo esencial y refleja los elementos de mayor relevancia sin aportar ideas nuevas.

Existen diversos tipos de resúmenes y uno de ellos es el extracto, el cual se forma a base de frases "extraídas" del propio texto [19,21]. La efectividad de las técnicas de resumen depende del siguiente fundamento: el resumen más apegado al documento original será aquel que sea desarrollado por el propio autor [21]. La extracción de información consiste en la creación de una representación estructurada, a partir de información seleccionada de un conjunto de textos. Actualmente los usuarios de

Internet desean obtener información lo más rápido posible, no desperdiciar su tiempo teniendo que revisar una larga lista de links, que probablemente no sean de su interés. Por ello uno de los objetivos de este trabajo se centra en exponer una metodología [13, 14] para generar extractos de documentos. Para conformar el extracto se siguió la siguiente metodología:

- Conversión de minúsculas: El documento a analizar tiene que ser sometido a un "pre-procesamiento", éste consiste en omitir la combinación de letras mayúsculas y minúsculas y, utilizar solamente letras minúsculas. De esta forma se facilitará el posterior tratamiento del documento y se eliminarán problemas de sensitividad [6].
- Identificación de tokens y unidad de extracción: Los signos de puntuación operan como delimitadores del lenguaje, y pueden indicar una pausa (la coma o el punto y coma), expresan inicio de una nueva idea (el punto), una aclaración (los paréntesis), etc. En un ambiente de programación los tokens realizan la misma función que los signos de puntuación, permiten dividir una cadena de caracteres en una serie de elementos delimitados por unos determinados caracteres. Este trabajo de investigación emplea como delimitador el punto (.), ya que de esta forma se segmentará el documento en oraciones o frases completas, las cuales fungirán a partir de este momento como la unidad de extracción, debido a que una oración comprende la expresión de una nueva idea de forma coherente y evita problemas de ambigüedad.
- Heurísticas para generación de extractos [21]:
  Heurística de Posición: Esta heurística consiste básicamente en dar mayor valoración a las primeras frases u oraciones de un texto. En dominios periodísticos, el título y las primeras frases de un texto dan una idea aproximada al lector del contenido del texto que se va a leer a continuación. Por esta razón se seleccionan las N primeras frases del documento que se está resumiendo. En este caso, el sistema opera con el valor típico de N el cual se maneja como cinco, es decir, el extracto contendrá como máximo cinco oraciones las cuales se consideran suficientes para representar el extracto de un documento.
  Heurística de Palabra Clave: Esta heurística consiste en extraer las frases donde se encuentra la palabra clave, la cual representa una *C* consulta que el usuario realiza al sistema. Es así como se seleccionan únicamente aquellas oraciones que podrían ser de interés al lector.

Para enriquecer el contenido del extracto se recurrió a utilizar ambas heurísticas de la siguiente forma: *Extraer las 5 primeras frases del documento que contengan la palabra clave.* El algoritmo [13] que ilustra lo anteriormente descrito se muestra a continuación:

Paso 1. Abrir documento para su lectura.
Paso 2. Conversión a minúsculas de todo el documento.
Paso 3. Segmentar el documento en oraciones completas delimitadas por (.)
Paso 4. Para cada oración identificar si se encuentra la palabra de consulta
  Si palabra consulta está contenida en oración
  Incrementar contador de frase de extracto(N)
  Almacenar tupla en la estructura hasta que N<=5
  Si no repetir paso 4.
Paso 5. Fin

El hilo del objeto activo selecciona alternadamente un método en la cola pendiente de peticiones y lo ejecuta. Por el lado del subsistema, este envía un llamado al OA, este es representado por un *proxy,* el cual tiene como principal responsabilidad generar objetos futuros para representar valores futuros, transformar llamados en objetos Request y ejecutar su copia profunda de objetos pasivos, pasados como parámetros [2].

ProActive proporciona una forma de mover un OA desde cualquier máquina virtual de Java a otra, dicha características se realiza a través de la primitiva migrateTo(...)[2, 28]. (Ver Tabla 1.)

Table 1. Primitivas de Migración en ProActive.

	Migración hacia:
migrateTo(URL)	Un Nodo Virtual identificado por un URL
migrateTo(Object)	La ubicación de algún OA

ProActive provee tres mecanismos para mantener comunicación con objetos móviles, el primero refiere a ubicar un servidor el cual guardará el rastro de los objetos móviles en el sistema. El segundo emplea una técnica completamente descentralizada conocida como "forwarders"; cuando se abandona un sitio, un OA deja un objeto especial que llamó un forwarder el cual apunta a su nueva ubicación. Hasta que se recibe un mensaje, este es pasado por el forwarder al objeto (otro forwarder). El tercero es un esquema original basado en los dos mecanismos anteriores y provee tolerancia a fallos [2].

Un OA será desplegado en varios ambientes heterogéneos donde las políticas de seguridad podrían diferir de un lugar a otro, al igual que el cómputo y las comunicaciones varían de host a host. Como tal las ubicaciones de OA efectivos no deben estar atadas al código fuente [2].

El primer principio es eliminar totalmente del código fuente los siguientes elementos [2, 28]:

- Nombres de máquinas
- Creación de protocolos
- Registro y búsqueda de protocolos

El objetivo de esto es desplegar cualquier aplicación en cualquier lugar sin cambiar el código fuente. Se pueden utilizar a su vez diversos protocolos como rsh, ssh, Globus y LSF para la creación de MVJ's necesarias para la aplicación.

Para alcanzar su objetivo, el modelo de programación cuenta con una noción específica de Nodos Virtuales (VN's) [2, 28]:

- Un VN es identificado como un nombre (un simple String),
- Un VN es usado en un programa fuente,
- Un VN es definido y configurado en un archivo descriptor (XML)
- Un VN, después de su activación, es mapeado a uno o a un conjunto actual de Nodos ProActive.

Obviamente, entidades distribuidas (OA), son creados en nodos, no en nodos virtuales. Los VN's son mucho más ricos en abstracción ya que proveen mecanismos tales como un mapeo cíclico, además de la capacidad de describir y disparar los mapeos a un VN que genera la asignación de varias MVJ's.

Por otra parte, un nodo es un objeto que vive en una MVJ, organizando al OA, existe una correspondencia entre los VN's y los nodos: la función creada por el despliegue: el mapeo, el cual puede ser especificado en un descriptor XML. Las operaciones que pueden ser configuradas en un descriptor son las siguientes [2,13,28]:

- El mapeo de VN's a nodos y a MVJ's
- Crear y adquirir MVJ's
- Registrar y buscar VN's

5 Estrategias de lanzamiento de agentes

Los agentes móviles además llevan consigo una lista que determinará los saltos o host por visitar, a ésta se le conoce normalmente como Itinerario del agente [3, 5], además controla casos especiales tales como, qué sucede si un destino no existe, y siempre sabe a donde irá en el siguiente traslado. De igual forma permite el reuso de itinerarios ya que son guardados como agendas. La migración [13], por tanto es la esencia de los agentes móviles, y en este trabajo el enlace para lograr que un agente viaje de un host a otro se realiza por medio de un identificador, en este caso es una dirección IP. Sin embargo el problema de migración es un problema NP-duro en el que se pretende optimizar las rutas de migración de agentes y minimizar los costos totales de comunicación [5, 11, 24].

El sistema es modelado usando agentes del tipo BDI (Beliefs, Desires, Intentions), es decir, Creencias, Deseos e Intenciones [5, 7, 10, 11]. Estas son abstraídas como clases que comprenderán la estructura del agente. Las creencias constituyen el conocimiento del agente sobre sí mismo y su ambiente, en este caso Internet. Los deseos son los objetivos que un agente tiene que cumplir, y para ello se vale de planes que materializan esos deseos y la convierten en una intención, la cual consistirá en generar extractos de documentos.

Para que un agente móvil pueda transitar por una red, es necesario que la plataforma esté apoyada sobre una infraestructura de comunicación determinada, de tal forma que sea dicha infraestructura la que realice la transferencia del agente de una máquina a otra. Las infraestructuras más habituales que utilizan los sistemas de agentes móviles son: TCP/IP o plataformas de más alto nivel como RMI (Remote Method Invocation).

ProActive actualmente utiliza la librería estándar RMI como capa de transporte portable, además trabaja sobre RMI para registrar y encontrar nodos. Un OA siempre es agregado a un nodo el cual representa una entidad lógica en una MVJ. Cuando se crea el OA se le tiene que asociar una URL o una referencia al nodo y este tiene que existir al momento de crear el OA, el cual es lanzado en una MVJ local o remota. [13,16,28]

Para que pueda ser accedido desde una MVJ remota, un nodo automáticamente registra con un RMI registry en la máquina local, la clase que realiza este procedimiento es NodeFactory en el método getNode [28].

Teniendo en cuenta las consideraciones en cómo es que los OA migran, en este trabajo se pretende implementar dos estrategias de generación de OA para extracción de texto en diferentes nodos, a fin de comparar el desempeño respecto al tiempo que son obtenidos los resúmenes.

Como se mencionó anteriormente se diseñó e implementó una clase que busca todos los documentos html almacenados en un host, una vez que los localiza, son abiertos para su lectura y se genera el resumen correspondiente de cada documento, este comportamiento es definido en el método runActivity de la interfaz RunActive, en la cual se especifica la actividad del OA [13, 28].

```
public void runActivity(Body body){
Service service=new Service(body);
   while(body.isActive())
  {  archihtml.buscando();
     service.blockingServeOldest();}}
```

5.1 Primera estrategia de lanzamiento

Se tiene como nodo raíz un host, el cual albergará una Base Datos que guarda extractos de cada documento, a su vez este mismo host fungirá como servidor lanzará diversos OA's, cada uno de ellos representa una dirección IP dentro de la local o bien algún nodo del clúster, es decir, nodos hijos. Posteriormente cada OA creado revisará las diferentes páginas. Cada vez que un OA clonado termina revisar la página, actualiza su información en un arreglo de clase, donde el almacena la dirección URL y el resumen extraído. Una vez que todos los objetos clonados han mandado la información obtenida a su nodo padre o servidor; este actualiza la Base Datos (Ver Figura 1.) [13].

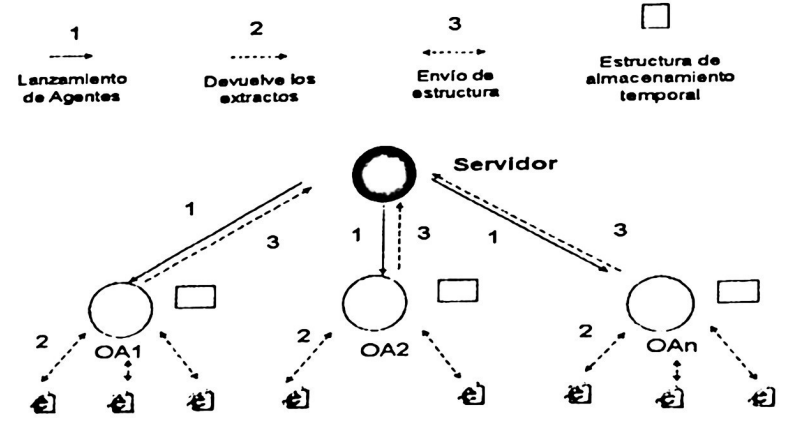

Fig. 1. Primera Estrategia de Lanzamiento. El número indica el orden en que se desarrolla actividad de los agentes.

En la clase Agente se definió el método mover, y como parte de su implementación se hizo uso de la primitiva migrateTo(String Node URL).

```
public void mover(String nodeURL) throws Exception{
  System.out.println(" Voy a migrar");
  ProActive.migrateTo(nodeURL);}
```

En este caso mediante el método mover(...), migra el OA cuyo hilo activo es llamado por este método a el nodo caracterizado por la URL proporcionada. Este método debe ser llamado desde un OA usando su hilo activo como el hilo actual a ser usado para localizar el llamado del OA al método. Es importante remarcar que el parámetro de la URL corresponde a una URL de un nodo existente.

Para especificar el itinerario del agente ProActive, a través de un archivo descriptor XML se detallan las características para definir nodos virtuales, es decir, se estable el número de nodos virtuales y el protocolo de comunicación entre los host [13].

```
<processDefinition id="rloginProcess">
<rloginProcess
class="org.objectweb.proactive.core.process.lsf.RLoginP
rocess" hostname="192.168.0.13">
<processReference refid="localJVM" />
</rloginProcess>
</processDefinition>
<processDefinition id="rloginProcess">
<rloginProcess
class="org.objectweb.proactive.core.process.lsf.RLoginP
rocess" hostname="192.168.0.14">
processReference refid="localJVM" />
</rloginProcess>
</processDefinition>
...
```

5.2 Segunda estrategia de lanzamiento

Se tiene la misma estructura de árbol mostrada anteriormente, con las siguientes excepciones; los objetos no serán lanzados de forma paralela, sino de uno en uno, una vez que se ha extraído la información de cada página HTML, regresará al servidor con su estructura para a actualizar la Base de datos, para esto no es necesario contar con un itinerario, es suficiente con especificar la dirección IP del destino al que se desea migrar [13].

```
org.objectweb.proactive.examples.Migrar.MigrarAgente
//remotehost/node
```

6 Resultados

En lo que concierne a ProActive fueron identificadas algunas restricciones que limitan su desempeño, las cuales se citan a continuación:

- Es necesario utilizar la misma versión de Java en cada uno de los host.
- Es necesario instalar la misma versión de ProActive en cada uno de los host.
- Es necesario que las aplicaciones desarrolladas en ProActive sean ejecutadas ambientes donde se tengan los permisos correspondientes, es decir, acceso determinada LAN, Grid o clúster.
- Es necesario para el lanzamiento de agentes que los host posean un protocolo establecido, ya sea rsh, rlogin, ssh, entre otros.

Sin embargo sus fortalezas radican en la facilidad para especificar protocolos comunicación sin la necesidad de realizar cambios al código fuente. Son una gran cantidad de protocolos los que se pueden emplear y actualmente conforme aparecen nuevas versiones se anexan más protocolos, uno de ellos es SSH y la ventaja proporcionar este tipo de servicio se ve reflejado en cuestiones de seguridad, problema grave de los agentes móviles, pues mediante este protocolo se pueden enviar datos en redes donde no se tiene certeza que sean del todo seguras.

Otra de las ventajas de ProActive son sus requerimientos mínimos, comparados con las bondades obtenidas con su uso, además funciona en sistemas operativos Windows y Unix. Debido a que ProActive posee características multiplataforma pruebas fueron realizadas en el sistema operativo Windows XP y Suse.

Hasta el momento se han implementado las dos estrategias de migración propuestas en este trabajo, las cuales han sido probadas sobre un cluster de 16 nodos donde cada nodo cuenta con 2 procesadores. En estos momentos sólo se están recuperando los resúmenes de las páginas html que se encuentran en el nodo servidor. Uno de los problemas a los que nos hemos enfrentado es que muchos hosts identifican a los agentes móviles como virus por lo cual la segunda estrategia aún se encuentra fase de prueba hasta lograr que los agentes tengan permisos de acceso.

En lo que respecta a la obtención de los extractos se ha presentado una herramienta para obtener las oraciones representativas de un texto, las cuales constituyen lo que llamado un extracto de texto. Las pruebas realizadas hasta el momento han sido realizadas con éxito. Se han revisado todas las páginas html que se encuentran en nodo local obteniéndose resúmenes usando pocos recursos lingüísticos y que brindan una representación cercana a los documentos originales.

7 Conclusiones y recomendaciones

Actualmente la importancia y la necesidad de implantar herramientas en sistemas distribuidos se ha acrecentado considerablemente, esto se ve reflejado en las diversas tecnologías desarrolladas para mantener comunicados equipos remotos que ejecuten tareas y tengan acceso a la información.

Una de las respuestas a este problema es ProActive, que es un API en constante crecimiento y en este momento se encuentra en su versión 2.1 por lo cual cada vez son más las habilidades que proporciona.

A pesar de que ProActive no ofrece herramientas para desarrollar aplicaciones con agentes móviles, con un manejo adecuado de los OA's se pueden crear estas entidades. Es importante considerar que esta API necesita extensiones de vital relevancia para su crecimiento, tales como una infraestructura genérica para agentes móviles, que agregue aspectos de seguridad y responda a situaciones que pudieran causar algún fallo, como por ejemplo si un agente no pudiese entrar a determinado host cómo se tendría que responder.

A su vez el hecho de ligar a los agentes móviles con el área de recuperación de información enriquece el trabajo sobremanera, de tal forma que la tarea de generar extractos de documentos se convierte en uno de los objetivos primordiales del agente (a parte de su migración). El algoritmo presentado sólo muestra las primeras 5 ocurrencias de la palabra de consulta contenida en algún documento, es decir, no analiza el valor informativo que pudiesen aportar otras oraciones. Por el momento no son revisados los contenidos, sólo se obtienen los resúmenes mediante los cuales el usuario decide consultar el documento original.

Una recomendación aunada a lo anteriormente expuesto es el diseño de un agente que revise la información obtenida y decida cual es la que se acerca más a lo que el usuario necesita.

Referencias

1. Arasu, A., Junghoo C., Garcia-Molina H., Paepcke A., Raghavan S.: Searching the Web, ACM Transactions on Internet Technology, Vol. 1, No. 1, August 2001, 2-43.
2. Baduel, L., Baude, F., Caromel, D., Contes, A., Huet, F., Morel, M., Quilici, R. ProActive: Programming, Composin, Deploying for the Grid, OASIS - Joint Project CNRS / INRIA / University of Nice Sophia – Antipolis, 2004, 12-38.
3. Baek, J.W., Kim, G.T., Yeom, H.Y.: Timed Mobile Agent Planning for Distributed Information Retrieval, Proceedings of the Fifth International Conference on Autonomous Agents, ACM Press, 2001, 120-121.
4. Binder, W., Roth, V.: Secure mobile agent systems using Java: where are we heading? Proceedings of the 2002 ACM symposium on Applied computing, ACM Press, 2002, 115-119.
5. Busetta, P., Ramamohanarao, K.: An architecture for Mobile BDI Agents, Proceedings of the 1998 ACM Symposium on Applied Computing, ACM Press, 1998. 445-452.
6. Gilbert, D., Aparicio, M., Atkinson, B., Brady, S., Ciccarino, J., Grosof, B., O'Connor, P., Osijek, D., Pritko, S., Spagna, R., Wilson, L.: IBM Intellitent Agents Strategy, IBM Corporation, 1995.
7. Hagimont, D., Ismail, L.: A Performance Evaluation of the Mobile Agent Paradigm, Proceedings of the 14th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages and Applications, ACM Press, 1999. 306-311.
8. Horvat, D., Cvetković, D., Milutinović, V., Koćović, P., Kovaćevic, V.: Mobile Agents and Java Mobile Agents Toolkits, Proceedings of the 33rd Hawaii International Conference on System Sciences, USA, Enero 2000,. 1-10.
9. Kobayashi, M., Takeda, K.: Information Retrieval on the Web, ACM Computing Surveys, Vol. 32, No, 2, June 2000. 144-173.

10. Komiya, T., Ohsida, H., Takizawa, M.: Mobile Agent Model for Distributed Systems, Proceedings of 22nd International Conference on Distributed Computing Systems Workshops, 2002. 131-136.
11. Kotz, D., Gray, R., Rus, D.: Transportable Agents Support Worldwide Applications, Proceedings of the 7th Workshop on ACM SIGOPS European Workshop: Systems Support for Worldwide Applications, ACM Press, 1996. 41-48.
12. Kotz, D., Gray, R.S.: Mobile Agents and the Future of the Internet, Department of Computer Science / Thayer School of Engineering Dartmouth College, In ACM Operating Systems Review 33(3), August 1999. 7-13.
13. Leal, L.:Tesis de Licenciatura: "Agentes Móviles para Generar Extractos de Documentos" Universidad Popular Autónoma del Estado de Puebla, Escuela de Ingeniería en Computación, Febrero 2005. 38-66, 73, 79-83, 90-107.
14. Leal, L., Vilariño, D., López, F.: Diseño de Agentes Móviles para Generar Extractos de Documentos en ProActive, Memorias del 2° Congreso Nacional de Ciencias de la Computación, Noviembre 2004, 287-292, ISBN 968 863 798 X.
15. Lu, J.: Some Research on Componentware Frameworks Base on Mobile Agent Technology, ACM SIGSOFT Software Engineering Notes Vol. 29 No. 2, March 2004, pp. 8-8.
16. Márquez, M.: Tesis de Maestría: "Infraestructura Genérica para Agentes Móviles para Aplicaciones Distribuidas bajo ProActive", BUAP Facultad de Ciencias de la Computación, Mayo 2004. 46-58.
17. Nolan, B.: Java and Information Retrieval from the Internet, Proceedings of the 2nd international Conference on Principles and Practice of Programming in Java, Computer Science Press, Inc., 2003. 133-135.
18. Pozo, S., Gasca, R.M., Gómez López, M.T.: Secure Tunnels for Mobile-Agent Systems, Proceedings of the Agent Technology Applications, 5th. Iberoamerican Workshop on Multi-Agent Systems, Iberagents 2004, Noviembre 2004. 12-25.
19. Proux, D., Rechenmann, F., Julliard, L.: A Pragmatic Information Extraction System, Proceedings of Seventh International Symposium on String Processing and Information Retrieval, 2000. 236-241.
20. Russel, S., Norving, P.:Artificial Intelligence A modern Approach, Second Edition, Ed. Prentice Hall Series in Artificial Intelligence, 2003, pp. 1-30,42-57, 840-850.
21. Salazar, H.: Tesis de Maestría: "Obtención de Extractos de Textos con base en un Corpus", Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Computación, Mayo 2004.1-27.
22. Shoeman, M., Cloete, E.: Architectural Components of Mobile Agent Systems, Proceedings of the 2003 Annual research conference of the South African Institute of Computer Scientists and Information Technologists on Enablement through Technology, South African Institute for Computer Scientists and Information Technologists, 2003. 48-58.
23. Weiss, G.: Multiagent Systems A Modern Approach to Distributed Artificial Intelligence, Massachusetts Institute of Technology, 1999, ISBN 0-262-23203. 1-77.
24. Xu, B., Lian, W. Giang, G.: Migration of Enterprise JavaBeans with ProActive, ACM SIGPLAN Notices, Vol. 38, No. 8, August 2003. 22-28.
25. Yanxiang, H., Yifeng C.: A GA- Based Solution to the Migration Problem of Mobile Agents in Distributed Information Retrieval Systems, Proceedings. 23rd International Conference on Distributed Computing Systems Workshops, Mayo 2003. 466-471.
26. http://www.fipa.org
27. http://www.omg.org
28. http://www.sop-inria.fr/oasis/ProActive