Aplicación de la programación distribuida en la obtención de patrones en la navegación por Internet de los usuarios de la UCI.

Husseyn Despaigne Reyes
Universidad de las Ciencias Informáticas, La Habana, Cuba, hdespaigne@uci.cu

Julio Antonio Hernández Pérez
Universidad de las Ciencias Informáticas, La Habana, Cuba, jhperez@uci.cu

Yoanni Ordoñes Leyva
Universidad de las Ciencias Informáticas, La Habana, Cuba, yordones@uci.cu

RESUMEN

Este trabajo persigue como objetivo principal reducir el tiempo que demora el procesamiento de los registros de navegación generados por el servidor *proxy* de la Universidad de las Ciencias Informáticas (UCI) en una herramienta informática, que automatizando un Proceso de Descubrimiento de Conocimientos en Base de Datos, es capaz de descubrir patrones de comportamiento de los usuarios al utilizar su cuota de navegación por Internet. Para ello se tienen varias fuentes de datos, siendo los *logs* la de mayor tamaño (hasta 30 GB en tan solo un mes). En tal sentido, se utilizó la programación distribuida con el *middleware* ICE (*Internet Communication Engine*) para analizar los datos de navegación, permitiendo que se pudiera manejar cuándo y cómo distribuir el procesamiento de los registros en dependencia de la cantidad de computadoras con las que se contaba, lo cual redujo el tiempo que demoraba realizar esta tarea y demostró que es una buena opción cuando se tiene que procesar grandes volúmenes de información en computadoras con no muy altas prestaciones. ICE constituyó una solución eficiente en ancho de banda, en uso de memoria, y en carga de CPU, garantizando flexibilidad y buen rendimiento. Esta investigación puede ser aplicada en un entorno similar.

Palabras claves: minería de datos, patrones de comportamiento, programación distribuida, registros de navegación.

ABSTRACT

This work pursues as its main objective to reduce the time it takes for the processing of navigational records generated by the proxy server of the University of Informatics Sciences (UCI) in a software tool that automating a process of Knowledge Discovery in Databases and is able to discover patterns of user behavior when using their navigation account for Internet browsing. This software will have several sources of data, where navigational records being the largest (up to 30 GB in just one month). In this regard, was used distributed computing with the middleware ICE (Internet Communication Engine) to analyze navigation data, allowing manage when and how to distribute the processing of records depending on the number of computers with which count, which reduced the time delayed to perform this task and proved to be a good choice when you have to process large volumes of information on computers with not very high performance. ICE established an efficient bandwidth, in memory usage and CPU load, ensuring flexibility and performance. This research can be applied in a similar environment.

Keywords: data mining, behavior patterns, distributed computing, navigational records.

1. INTRODUCCIÓN

El cursar de los años y el desarrollo de las tecnologías han generado un crecimiento considerable de datos (Hernández et al., 2004). El uso de esta información para el perfeccionamiento de los procesos de las empresas e instituciones crea la necesidad de desarrollar nuevas técnicas y herramientas para analizar esta enorme cantidad de datos (Olmos y González, 2007), siendo la Minería de Datos (MD) una de estas técnicas. Larose plantea que la MD: "... es el proceso de descubrir nuevas correlaciones significativas, patrones y tendencias ocultas a través de grandes cantidades de datos almacenados en los repositorios, utilizando tecnologías de reconocimiento de patrones, así como técnicas estadísticas y matemáticas" (Larose, 2005).

Uno de los escenarios donde se ha aplicado la MD es la World Wide Web (WWW). Su acelerado crecimiento y la competencia entre las organizaciones han traído la necesidad de mejorar la calidad de los servicios que se brindan en Internet, utilizando como base el comportamiento de los usuarios que los usan. Se le ha denominado Minería Web (MW) al descubrimiento de información útil en la WWW. La MW posee varias clasificaciones atendiendo al contenido que se analiza, una de ellas es la Minería de Uso de la Web (Web Usage Mining: WUM) centrada en el análisis de los *logs* de servidores *web*, servidores *proxy*, etc. Los servidores *proxy*, encargados de gestionar el acceso a Internet de algunas instituciones, generan un alto volumen de *logs*, que archivan la información relativa a la navegación de un grupo de usuarios de una determinada organización.

La UCI cuenta con un servicio de navegación por Internet para miles de usuarios y posee un insuficiente ancho de banda para brindar un buen servicio. Esta situación provoca que la navegación sea lenta, además no todos los usuarios hacen un uso de Internet acorde a los intereses de la Universidad, no existiendo ningún mecanismo que diferencie en cuestiones de calidad de servicio a los usuarios que realizan una navegación enmarcada en los intereses de la institución y los que no lo hacen así. Para la Dirección de Redes y Seguridad Informática (DRSI), encargada de garantizar el correcto funcionamiento de la red de computadoras en la UCI, puede resultar interesante encontrar patrones que describan el comportamiento de los usuarios al usar su cuota de navegación por Internet, permitiendo mejorar la toma de decisiones en cuanto al sistema de cuotas de navegación implementado en la Universidad. Para ello se debe procesar un gran volumen de información proveniente de distintas fuentes, siendo los registros de navegación almacenados por el servidor *proxy* los de mayor tamaño. El análisis de grandes volúmenes de datos es engorroso realizarlo manualmente, siendo necesaria su automatización, por tanto, es conveniente utilizar técnicas de MD para este propósito.

Con este fin se desarrolló una Herramienta Informática de Minería de Uso de la Web aplicada a los registros de navegación por Internet (HERMINWEB). Esta herramienta fue desarrollada en el Centro de Telemática de la UCI por los autores del presente trabajo y otros autores y fue el fruto de una investigación para una tesis de maestría en Ciencias de la Computación y dos tesis de pregrado para optar por el título de Ingeniero en Ciencias Informática que otorga la UCI. HERMINWEB automatiza un proceso KDD, dentro del cual además de otras, se insertan la MD y la Preparación de los Datos (PD) como fases, siendo esta última la que mayor costo computacional requiere. Durante la PD se procesan los registros de navegación del *proxy*. En la primera versión de HERMINWEB se realizaba el procesamiento de los *logs* basado en la programación concurrente utilizando varios hilos de ejecución. Esta solución mejoraba el tiempo en cuanto a la variante de hacerlo de manera secuencial, pero aún demoraba el proceso un tiempo considerablemente alto teniendo en cuenta que para procesar 30 GB de *logs* se consumían alrededor de 2 horas e incluso más.

De acuerdo a la situación descrita se tiene el siguiente problema científico ¿Cómo reducir el tiempo de procesamiento de los *logs* del *proxy* durante la fase de Preparación de los Datos en la obtención de patrones que describan el comportamiento de los usuarios de la UCI al utilizar su cuota de navegación por Internet? El objetivo de la presente investigación es reducir el tiempo de procesamiento de los *logs* del *proxy* durante la fase de Preparación de los Datos en la obtención de patrones que describan el comportamiento de los usuarios de la UCI al utilizar su cuota de navegación por Internet.

El presente trabajo muestra el desarrollo de una solución basada en la programación distribuida que permite reducir el tiempo de procesamiento de los registros de navegación del *proxy* durante la PD, la cual forma parte de

un proceso KDD automatizado por la herramienta HERMINWEB para obtener patrones en la navegación por Internet de los usuarios de la UCI. Dicho proceso es guiado por la metodología CRISP-DM (Chapman et al., 2000). Para realizar la programación distribuida se utilizó el *middleware*[1] *Internet Communication Engine*[2] (ICE) (Vallejo, 2006).

2. DESARROLLO

2.1 EL PROCESO DE DESCUBRIR CONOCIMIENTOS EN BASES DE DATOS

El término Minería de Datos en muchas ocasiones se utiliza como sinónimo de Descubrimiento de Conocimiento en Bases de Datos, siendo en realidad la MD una de las fases por las que está compuesto el proceso KDD, según la metodología CRISP-DM es la llamada *Modelado*. CRISP-DM define seis fases para un proceso KDD (Chapman et al., 2000): *Comprensión del negocio, Comprensión de los Datos, Preparación de los Datos, Modelado, Evaluación, Implantación*.

Primeramente es imprescindible comprender a profundidad el dominio donde se desenvuelve la investigación. En la PD se determinan las fuentes de información que pueden ser útiles; se transforman los datos a un formato común, se detectan y se resuelven las inconsistencias presentes en los mismos, se eliminan o corrigen los datos incorrectos; además, se consideran únicamente aquellos atributos que van a ser relevantes. En la fase *Modelado* se aplica el modelo, la tarea, la técnica y el algoritmo seleccionado para la obtención de reglas y patrones. Luego en la fase de *Evaluación* se comprueban los patrones y se analizan por expertos, se puede regresar a la fase *Análisis del Problema* en caso de querer perfeccionar los resultados. Finalmente, en la fase de *Explotación* se comparte el nuevo conocimiento con los interesados. Las fases que componen el KDD hacen que su desarrollo sea un proceso iterativo e interactivo con el usuario.

2.2 MINERÍA DE DATOS APLICADA A LOS REGISTROS DE NAVEGACIÓN EN LA UCI

La UCI cuenta con un servicio de navegación por Internet para miles de usuarios. Para ello cuenta con servidores *proxy* que gestionan todo el flujo de peticiones realizadas. Los sistemas actualmente instalados y en explotación en la DRSI (Martín y García, 2007) no cubren todo el conocimiento implícito en los registros de navegación, dificultando la toma de decisiones a la DRSI. Por otra parte se cuenta con sistemas de gestión de información de los trabajadores y gestión académica de los estudiantes. Esta información en conjunto con los registros de navegación puede ser de mucha utilidad e interés para la DRSI, con ella se pueden encontrar patrones que describan el uso de la navegación de los diferentes usuarios de la institución.

Con tal propósito se desarrolló una herramienta capaz de: obtener y mezclar los datos registrados por el servidor *proxy* con los datos de los usuarios almacenados en los sistemas que gestionan la información de los trabajadores y estudiantes; así como, extraer patrones descriptivos presentes en los datos, enfocados en las tareas de agrupamiento y reglas de asociación, con el fin de encontrar clases de usuarios que se comporten de manera similar en el uso de la cuota navegación y relaciones entre los atributos de estas clases. La herramienta fue desarrollada en el Centro de Telemática (TLM) de la UCI y está escrita en el lenguaje de programación *Python*.

2.3 EL MIDDLEWARE INTERNET COMMUNICATION ENGINE

ICE es un middleware orientado a objetos, es decir, ICE proporciona herramientas, APIs, y soporte de bibliotecas para construir aplicaciones cliente-servidor orientadas a objetos. Una aplicación ICE se puede usar en entornos heterogéneos: los clientes y los servidores pueden escribirse en diferentes lenguajes de programación, pueden ejecutarse en distintos sistemas operativos y en distintas arquitecturas, y pueden comunicarse empleando

[1] Software de conectividad que hace posible que aplicaciones distribuidas puedan ejecutarse sobre plataformas con distintos sistemas operativos, que usan distintos protocolos de red y, que incluso, involucran distintos lenguajes de programación en la aplicación distribuida.
[2] En español: Motor de Comunicaciones por *Internet*.

diferentes tecnologías de red. Además, el código fuente de estas aplicaciones puede portarse de manera independiente al entorno de desarrollo. Cada objeto ICE tiene una interfaz con un determinado número de operaciones. Las interfaces, las operaciones, y los tipos de datos intercambiados entre el cliente y el servidor se definen utilizando el lenguaje *Slice*[3], el cual permite definir el contrato entre el cliente y el servidor independientemente del lenguaje de programación empleado (Vallejo, 2006).

2.4 PREPARACIÓN DE DATOS EN HERMINWEB

Para realizar la PD se utilizaron varias fuentes de información con el objetivo de mezclar los datos de la navegación con las características de los usuarios, para así obtener patrones descriptivos del uso de las cuotas de navegación. Dichas fuentes son las siguientes:

- Los registros de navegación del servidor *proxy*.
- Los sistemas de recursos humanos pertenecientes a los trabajadores (son dos servicios web, uno para los trabajadores que residen en la Universidad y otro para los que brindan servicios en ella, en este documento se tratarán como trabajadores internos y externos respectivamente).
- El sistema de gestión académica estudiantil llamado Akademos.
- Las clasificaciones de sitios *web* de *Internet* usando *BlackLists*.

Los datos de los trabajadores son obtenidos mediante servicios *web* y los pertenecientes a los estudiantes accediendo directamente a la base de datos de Akademos, en este caso el gestor es *Microsoft SQLServer*. Al terminar este proceso se tienen los datos listos para ser analizados en la fase de *Minería*. La PD es una de las tareas que más procesamiento computacional necesita, pues se analiza un volumen elevado de información (la media de logs generados en un mes es 30 GigaBytes). En esta fase se procesan los *logs* del servidor *proxy* y se mezclan con los datos de los usuarios. Las computadoras que intervienen en la investigación cuentan con un CPU Intel Dual Core con 2.1 GHz de velocidad y 1.0 GB de memoria RAM, situación que requiere ser analizada para reducir lo más posible el costo en cuanto a recursos y tiempo.

Una de las cuestiones importantes en el desarrollo de aplicaciones es el tiempo de ejecución en la realización de las tareas. En tal sentido, fue descartado el procesamiento secuencial para analizar los *logs*. Una de las soluciones aplicadas fue el procesamiento basado en hilos; esta solución mejoró el tiempo de ejecución pero seguía siendo alto, para 30 GB de *logs* consumió 2 horas aproximadamente. Para ello se utilizaba la librería *pp*[4] de *Python*, pero la misma solo distribuía el trabajo cuando existía una carga considerablemente elevada; mientras no ocurriera esto, el cliente donde se estuviera ejecutando la herramienta sería el encargado de procesar todos los datos, impidiendo manejar cuándo y de qué forma distribuir el procesamiento de los *logs*.

Se tomó como alternativa la programación distribuida, siendo la definitiva a utilizar. Para el procesamiento de los *logs* se utilizó la implementación que ofrece *Python* para el *middleware* ICE en la librería *python-zeroc-ice* por ser eficiente en ancho de banda, en uso de memoria, y en carga de CPU[5], siendo una buena solución para este problema, además de que su uso garantiza que se pueda manejar cuándo y de qué forma distribuir el procesamiento de los registros en dependencia de la cantidad de nodos con las que se cuente. Se creó un *parser* distribuido en varias computadoras siendo HERMINWEB el Nodo Central y las demás computadoras los Nodos de Procesamiento (de aquí en adelante: servidores ICE). La herramienta distribuye los usuarios a cada nodo para que procese sus registros. Su diseño se muestra en la Figura 1.

[3] Del inglés: *Specification Language for Ice*. En español: Lenguaje de especificación para ICE.
[4] Del inglés: Parallel Pyhton. Librería utilizada para la programación paralela con Python.
[5] Del inglés: Central Processing Unit. En español: Unidad Central de Procesamiento.

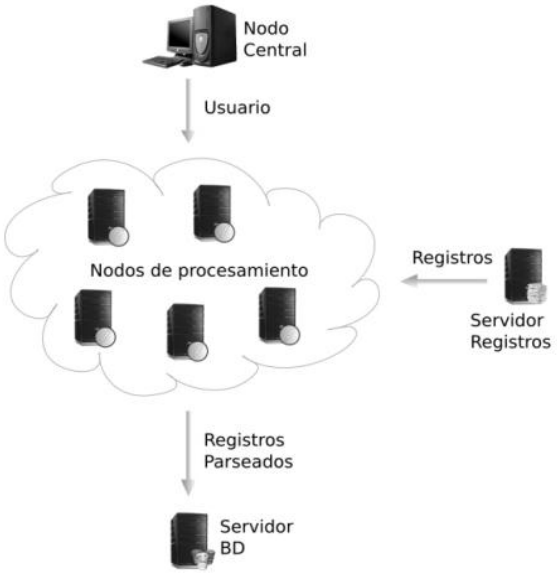

Figura 1: Distribución de los nodos de procesamiento

Se creó una pequeña aplicación independiente de HERMINWEB con el objetivo de publicar en un servidor ICE remoto el objeto que permita invocar desde el nodo central la función (*parsearUsuario*) definida para procesar los registros de los usuarios. Las librerías de *Python* necesarias para esta tarea son: *urllib2, cStringIO, re, time, psycopg2, calendar, os, base64, itertools*. Esta aplicación contiene como elementos fundamentales:

- Una capa de abstracción denominada *IntermediarioICE* que es responsable de implementar las funcionalidades necesarias para publicar un objeto en un servidor remoto, acceder al mismo desde un cliente y apagar el servidor.
- Un fichero llamado *ParserLogs.ice*, en el cual se define el método *parsearUsuario* perteneciente a la *interface Parser*, que forma parte del módulo *ParserLogs*. Este método será redefinido por la clase que implemente la *interface* e invocado desde un cliente utilizando el objeto publicado por el servidor. La declaración del método y el módulo está escrita en *Slice*. *parsearUsuario* retorna una cadena de caracteres informando del resultado del procesamiento de los *logs* del usuario y recibe como parámetros el nombre de usuario, las transformaciones realizadas a las variables de la navegación (horario, consumo, peticiones y rango *IP*), las credenciales de conexión al servidor web donde se encuentran publicados los *logs* (Servidor Registros) y al servidor de base de datos (Servidor BD), así como una cadena con todas las *urls* de acceso a los *logs* que se deben procesar para dicho usuario.
- Un fichero llamado *ParserLogs.py* donde primeramente se obtiene una instancia del módulo declarado en *ParserLogs.ice* utilizando el método *obtenerModulo* implementado en la clase *IntermediarioICE* de la capa de abstracción del mismo nombre, para lo cual se usa la sentencia:

```
Modulo = IntermediarioICE().obtenerModulo('ParserLogs', 'ParserLogs.ice')
```

En este fichero además se declara la clase *ParserLogs*, la cual debe implementar la *interface Parser*, redefiniendo el método *parsearUsuario*, en el cual para cada *log* se van obteniendo los datos de las conexiones que correspondan con la configuración establecida inicialmente en HERMINWEB y se van almacenando en una lista. Luego esta lista se ordena ascendentemente con el objetivo de agrupar las peticiones realizadas al servidor *proxy* según las variables seleccionadas para crear las vistas minables utilizando la librería *itertools* de Python y de esta forma crear las sesiones de usuario. Una sesión no es más que un perfil que se crea con el objetivo de agrupar el comportamiento de los usuarios al navegar por *Internet* tomando en cuenta diferentes variables como: horario de acceso, dirección *IP*, sitio accedido, etc. y de esta forma evitar inconsistencias e información redundante. Al

contar con muchas peticiones (alrededor de 130 millones en un mes) las consultas realizadas a la base de datos demoraban mucho tiempo. Creando las sesiones de usuario se redujo considerablemente la cantidad de datos. Finalmente las sesiones son almacenadas en una tabla temporal en la base de datos relacional y se retorna un mensaje con el resultado del proceso.

Otra de las funcionalidades importantes de esta clase es *publicar*, la cual recibe como parámetro el objeto que se va a publicar. En primer lugar se establece un punto de comunicación en el servidor, teniendo en cuenta el puerto, el identificador del servidor y el protocolo de comunicación. Luego se publica el objeto y se crea un nuevo hilo para esperar por las conexiones desde los clientes y se actualiza el estado del servidor como *Iniciado*. Igualmente cuenta con un método *apagar* que termina la espera del servidor y actualiza su estado como *Apagado*.

HERMINWEB actúa como un cliente que invoca los métodos en los servidores remotos utilizando los objetos publicados por estos últimos. Con este objetivo se implementó la clase *ManagerParserLogs*, la cual tiene como método esencial *parsearLogs*, en el cual primeramente se llena una cola de tareas por cada tipo de usuario (estudiantes, trabajadores internos y trabajadores externos). Cada elemento de la cola es una terna del tipo *NombreUsuario-Proxy-IPServidor*, donde *NombreUsuario* contiene el nombre de usuario del cual se deben *parsear* los *logs*; *Proxy* es un objeto publicado por un servidor ICE concreto y que es retornado por el método auxiliar *__obtenerProxy*; *IPServidor* no es más que una cadena con la dirección *IP* del servidor ICE al cual pertenece el *Proxy* obtenido. La estrategia utilizada para distribuir los usuarios a los distintos nodos de procesamiento fue la siguiente:

1. Para cada servidor ICE proporcionado en la lista de servidores (esta lista contiene las direcciones *IP* de todos los servidores ICE a utilizar) a partir de la configuración inicial se obtiene su correspondiente *Proxy* y se almacena en una lista de *proxys*. La computadora donde se encuentra ejecutándose HERMINWEB en este instante por defecto también se convierte en un servidor ICE con el objetivo de agilizar aún más el procesamiento de los datos y lógicamente también se obtiene un *Proxy* local y se almacena en la lista.
2. Cada usuario se va asignando a un *Proxy* en el orden en que estos últimos aparecen en la lista de manera tal que el trabajo sea lo más equitativo posible y esta información se pasa a la cola según el tipo de usuario como se explicó anteriormente.
3. Una vez llenas las colas de tareas se pasa a procesar los usuarios en varios hilos, invocando al método *parsearUsuario* de cada *Proxy*.
4. Finalmente se apaga el *proxy* local.

Vale destacar que esta estrategia no es óptima debido a que si un servidor es más rápido que otro procesando los datos puede caer en un estado de inanición mientras el otro en una sobrecarga de trabajo, desaprovechando la oportunidad de que se reduzca más el tiempo en el procesamiento de los datos. Una mejora en este sentido pudiera ser que cada vez que un nodo procese los *logs* de un usuario emita una señal que indique que está listo para procesar otro y en ese momento se le asigne la tarea. Por tanto el estudio de una mejor solución queda pendiente para próximas versiones de la herramienta. Las pruebas realizadas para el procesamiento de los *logs* de 12 313 usuarios arrojaron los resultados que se muestran en la Tabla 1.

Tabla 1: Tiempo requerido según cantidad de nodos

Número de nodos	Tiempo total aproximado (minutos)
1	86
2	43
3	30
4	15

3. CONCLUSIONES

Como resultado de la investigación se redujo el tiempo de procesamiento de los registros del *proxy* durante la fase de Preparación de los Datos en la obtención de patrones que describen el comportamiento de los usuarios de la UCI al utilizar su cuota de navegación por Internet. Para ello, la utilización de la programación distribuida demostró que es una excelente opción cuando se tiene que procesar grandes volúmenes de información en computadoras con no muy altas prestaciones, pues redujo considerablemente el tiempo de procesamiento de los registros de navegación. El uso del *middleware* ICE constituyó una solución más eficiente en ancho de banda, en uso de memoria, y en carga de CPU que la utilizada anteriormente con la librería *pp* de *Python*, garantizando mayor flexibilidad en cuanto a cuándo y de qué forma distribuir el procesamiento de los registros en dependencia de la cantidad de nodos con las que se cuente. No obstante resulta interesante seguir profundizando en el tema con el objetivo de disminuir más el tiempo de procesamiento de los *logs* en HERMINWEB. Para trabajos futuros se recomienda:

- Estudiar otras estrategias para distribuir los usuarios hacia los nodos de procesamiento que permitan reducir más el tiempo que demora procesar los *logs* en HERMINWEB.
- Estudiar las ventajas que proporciona ICE para ejecutar las llamadas a los procedimientos en un servidor remoto de manera asíncrona.
- Estudiar la posibilidad de que la publicación de los objetos remotos en los servidores ICE sea gestionada desde HERMINWEB y no en una aplicación independiente.

REFERENCIAS

Chapman, Pete, y otros. 2000. “CRISP-DM 1.0: Step-by-step data mining guide”. s.l. : SPSS Inc., 2000.

Hernández Orallo, José, Ramírez Quintana, María José y Ferri Ramírez, César. 2004. “Introducción a la Minería de Datos”. Madrid : Pearson Prentice Hall, 2004.

Larose, Daniel T. 2005. “Discovery Knowledge in Data: An Introduction to Data Mining”. New Jersey : Wiley, 2005.

Martín Álvarez, Luis Orlando y García Martínez, Yassier. 2007. “Sistema de Reportes de Navegación por Internet”. Unicersidad de las Ciencias Informáticas. 2007.

Olmos, I. y J, González. 2007. “Minería de Datos”. Puebla : s.n., 2007.

Vallejo Fernández, David. 2006. “Documentación de ZeroC ICE”. Universidad de Castilla-La Mancha. Castilla : s.n., 2006.

Autorización y Renuncia

Los autores autorizan a LACCEI para publicar el escrito en las memorias de la conferencia. LACCEI o los editores no son responsables ni por el contenido ni por las implicaciones de lo que esta expresado en el escrito