INVESTIGACIÓN Y CIENCIA
DE LA UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES
Número 41, (9-19), Mayo-Agosto 2008

Optimizando el balanceo dinámico de carga bajo CORBA en un sistema neuronal de verificación de firmas *off-line*

Dr. Francisco Javier Luna Rosas [1], Dr. Julio César Martínez Romo [1],
Dr. Jaime Muñoz Arteaga [2], Lic. Gricelda Medina Veloz [2]
y Dr. Miguel Mora Gonzaléz[3]

RESUMEN

El balanceo de carga es parte del amplio problema de asignación de recursos. Equilibrar carga significa distribuir los procesos entre los elementos de procesamiento para lograr algunas metas de desempeño tales como: el tiempo de ejecución, los retardos de comunicación, y/o maximizar la utilización de recursos. Este artículo propone una nueva forma de optimizar el tiempo de respuesta global en un sistema neuronal de verificación de firmas fuera de línea (*off-line*), basado en una arquitectura que opera en dos fases, la fase de entrenamiento y la de verificación. El sistema de verificación fue implementado en una arquitectura de balanceo dinámico de carga construida bajo el estándar de CORBA.

ABSTRACT

Load balancing is part of the extensive problem of resource assignment. The load balancing problem consists in distributing some processes among some processing elements in order to achieve some performance goals such as: minimizing the execution time, communication delays or maximizing the use of resources. This article proposes a new form of optimizing the total execution time in a neural system of *off-line* signature verification, based over an architecture that operates in two phases, a training phase and a verification phase. The verification system was implemented in a dynamic load balancing architecture which was built under the CORBA standard.

Palabras clave: Verificación de firmas fuera de línea, CORBA, balanceo dinámico de carga, redes neuronales.
Key Words: *Off-Line signature verification, CORBA, dynamic load balancig, neural network.*

Recibido: 30 de enero de 2008, aceptado: 7 de mayo de 2008

[1] Instituto Tecnológico de Aguascalientes, Av. Adolfo López Mateos 1801 Ote. Esq. Av. Tecnológico, Fracc. Ojocaliente, C.P. 20256, Aguascalientes, Ags.
[2] Universidad Autónoma de Aguascalientes, Departamento de Sistemas de Información, Av. Universidad 940, Aguascalientes Ags., C.P. 20100.
[3] Universidad de Guadalajara, Centro Universitario de Lagos, Departamento de Ciencias Exactas y Tecnologías.

INTRODUCCIÓN

Las estrategias de balanceo de carga pueden ser divididas en dos grupos grandes: Estrategias de balanceo estático y dinámico. Las primeras, obtienen la localización de todos sus procesos antes de comenzar la ejecución. Las estrategias de balanceo dinámico intentan equilibrar la carga en tiempo de ejecución (Shirazi, 1995).

Cuando se hace balanceo de carga se aplica una técnica bien establecida para utilizar los recursos de computación disponibles más efectivamente, las aplicaciones distribuidas pueden mejorar su escalabilidad, tiempo de respuesta y uso de recursos empleando balanceo de carga en varias formas y en varios niveles del sistema, incluyendo la red, el sistema operativo y a nivel *middleware* (Ossama, 2001b).

1) Balanceo de carga a nivel de red. Los Servidores de Nombres de Dominio (DNS) y los Ruteadores IP´s que sirven a una gran cantidad de máquinas *host* proveen este tipo de balanceo de carga (Cisco, 2000).

ISSN 1665-4412

INVESTIGACIÓN Y CIENCIA
DE LA UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES
Número 41, (9-19), Mayo-Agosto 2008

2) Balanceo de carga a nivel del sistema operativo. Los sistemas operativos distribuidos generalmente proveen este tipo de balanceo de carga a través del agrupamiento de computadoras, compartición de carga y mecanismos de migración de procesos (Goscinski, 1992).

3) Balanceo de Carga Basado en *Middleware.* Las interacciones globales son acopladas por arquitecturas llamadas *middleware*. La arquitectura más común de *middleware* para aplicaciones orientadas a objetos distribuidas es *Common Object Request Broker Architecture* (CORBA) (OMG, 2001). Ha existido importantes enfoques para extender funcionalidades de CORBA que soporten balanceo de carga, por ejemplo, TAO (Ossama, 2001a), VisiBroker (Lindermeier, 2000), MICO (Puder, 2002), etc. Para consolidar este enfoque y resolver el problema, la OMG diseñó un *Request For Proposal* (RFP) (OMG, 2001), la cual es una propuesta para extender la funcionalidad de CORBA para balancear y monitorear carga en ambientes basados en CORBA, para procesar distribuidamente y con alto desempeño las aplicaciones implementadas en el estándar.

A continuación se analiza a detalle los conceptos claves de un servicio de balanceo de carga basado en CORBA Figura 1 (Ossama, 2001a):

Balanceador de carga. Es un componente que intenta distribuir la carga a través de grupos de servidores de una manera óptima. Un balanceador de carga debe consistir de un simple servidor centralizado o múltiples servidores descentralizados que colectivamente forman un balanceador lógico.

Réplica. Es un duplicado de un objeto particular sobre un servidor que es manejado por un balanceador de carga. Ésta ejecuta las mismas tareas que el objeto original.

Grupo de objetos. Es un grupo de réplicas a través del cual la carga es balanceada. Las réplicas en tal grupo implementan las mismas operaciones remotas.

Sesión. En el contexto de balanceo de carga *middleware*, una sesión se define como el periodo de tiempo que un cliente invoca operaciones remotas (requerimientos) para acceder servicios proporcionados por objetos en un servidor particular (Figura 1).

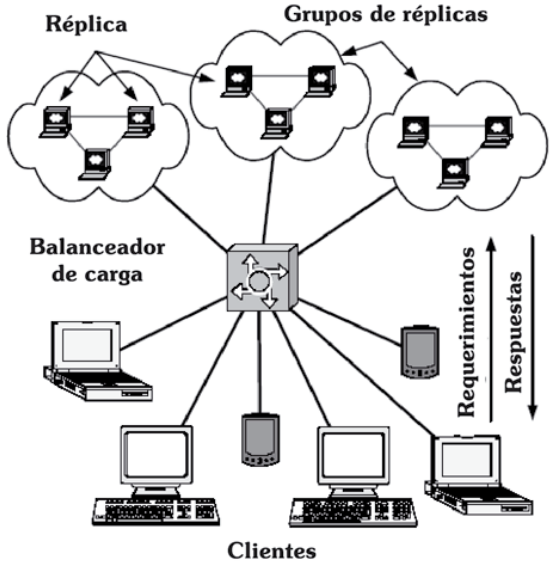

Figura 1. Conceptos claves en un servicio de balanceo de carga *Middleware* (IONA, 2002).

MATERIALES Y MÉTODOS

A. Diseño del servicio de balanceo de carga bajo CORBA

El proceso de desarrollo de *software* moderno incluye el diseño de patrones de *software* los cuales son una descripción formal de buenas soluciones a problemas ya planteados anteriormente, éstos pueden ser usados por los desarrolladores de *software* como una colección de conocimiento experto acerca de un problema específico. Un amplio rango de patrones de diseño puede ser obtenido, en (Buschman, 1996), (Gamma, 1995).

En esta sección se analiza la construcción de la arquitectura de balanceo de carga mediante patrones de *software* que fueron utilizados en la construcción y diseño de la misma, con ello se busca facilitar la futura reutilización del diseño y la arquitectura del sistema, logrando crear un lenguaje común de comunicación entre los desarrolladores, además de promover el uso de buenas prácticas en el proceso de diseño y construcción.

A.1 Patrón *Broker*

Este patrón presenta un conjunto de componentes desacoplados que interactúan a través de invocaciones a servicios remotos (Buschman, 1996). En el servicio de balanceo propuesto, representa el equilibrio que permite a los componentes acceder a los servicios que ofrecen otros

componentes mediante invocaciones de servicio remotas y transparentes a la localización de los servidores, para cambiar, añadir o eliminar componentes en tiempo de ejecución (Figura 2).

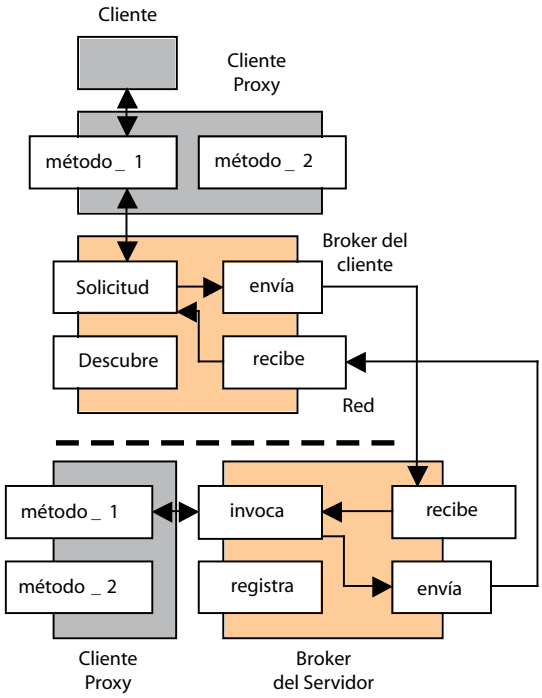

Figura 2. Patrón Broker.

En el balanceo, estos requerimientos son pasados a través de *proxies* al *broker*. El *broker* busca para su localización al objeto servidor correspondiente y le pasa los requerimientos.

A.2 Patrón Interceptor

Permite agregar fácilmente funcionalidad al sistema para cambiar su comportamiento dinámicamente, sin necesidad de recompilarlo, es decir, hace el cambio de comportamiento en tiempo de ejecución (Curry, 2004), (Douglass, 2000) ya sea interponiendo una nueva capa de procesamiento o cambiando el destino dinámicamente, ya que interpone objetos los cuales pueden interceptar llamadas e insertar un procesamiento específico que puede estar basado en el análisis del contenido, además de redireccionar una llamada a un punto diferente Figura 3 (Buschman, 2007).

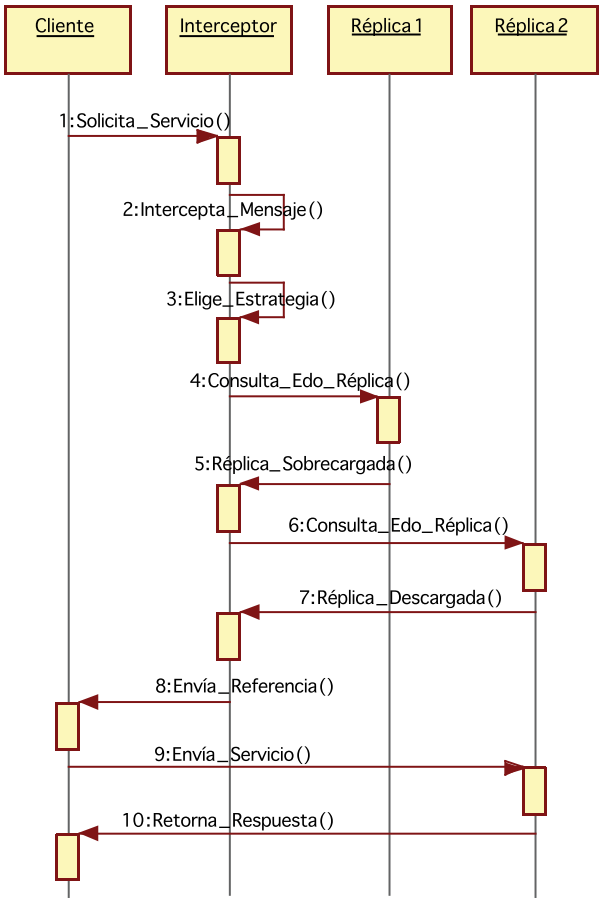

Figura 3. Diagrama de secuencia del patrón Interceptor.

En el servicio de balanceo, este patrón representa el mecanismo para redirigir los requerimientos de los clientes a una réplica apropiada. Las aplicaciones basadas en CORBA contienen las construcciones necesarias para soportar transparentemente el envío de requerimientos de los clientes (Douglass, 2000).

A.3 Patrón Estrategia

En general, no es común que todas las aplicaciones distribuidas exhiban las mismas condiciones de carga, significa que algunas estrategias de balanceo son más aplicables a algunos tipos de aplicaciones que a otras. Un balanceador de carga debe ser lo suficientemente flexible para soportar diferentes tipos de estrategias de balanceo. Figura 4, (Buschman, 1996), (Stelting, 2003).

INVESTIGACIÓN Y CIENCIA
DE LA UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES
Número 41, (9-19), Mayo-Agosto 2008

Las estrategias encapsuladas en este patrón son las siguientes:

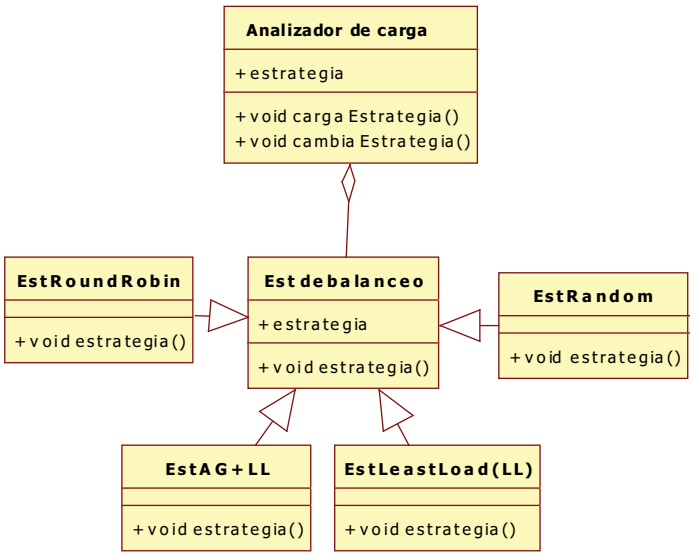

Figura 4. Diagrama de clases de patrón estrategia.

Least_Loaded (LL). La estrategia LL es utilizada para el balanceo dinámico de carga, a diferencia de la estrategia *Random* y *Round Robin*, la estrategia *LL* usa una estrategia de balanceo de carga adaptativa. Como su nombre lo indica esta estrategia elige dinámicamente al miembro de un grupo de objetos con la carga más baja utilizando información en tiempo de ejecución.
AG + LL. Esta estrategia combina un algoritmo genético (AG) en conjunto con *Least_Loaded* para balancear dinámicamente la carga, (Luna, 2003), (Luna 2005).

La mejor forma de resolver un problema de balanceo de carga es centrar el problema como un problema de optimización (Goscinski, 1992). En la última década, los Algoritmos Genéticos (AGs) han sido extensamente usados como herramientas de búsqueda y optimización en varios dominios de problemas, incluyendo las ciencias, el comercio y la ingeniería. La razón primaria de su éxito es la amplia aplicabilidad, facilidad de uso y perspectiva global (Goldberg, 1989).

Una explicación detallada acerca de la combinación de algoritmos genéticos con *Least_Loaded* para optimizar el tiempo de respuesta de una aplicación en el balanceo dinámico de carga está publicada en (Luna, 2003), (Luna 2005). Cabe hacer mención que en este trabajo se tomaron como base los resultados obtenidos por Braun Tracy D., Siegel H. J. en (Braun, 2001) y el trabajo de Albert Y. Zomaya en (Zomaya, 2001) donde demuestran cómo los algoritmos genéticos pueden ser una técnica adecuada para la planificación de tareas, mediante un análisis comparativo de algoritmos genéticos contra otras técnicas de optimización clásicas y heurísticas tales como: búsqueda tabú, recocido simulado, A*, *Oportunistic Load Balancing* (OLB), etc.

B. Arquitectura del sistema neuronal de verificación de firmas *Off-Line*

B.1 Panorama general de la verificación de firmas

Una firma puede ser verificada *on-line* y *off-line*. En el primer caso, una pluma instrumentada o una tableta digitalizadora es utilizada para capturar la forma de la firma y la dinámica del movimiento de la mano (Plamondon, 2000). La técnica *off-line* se refiere a situaciones en que la firma fue realizada previamente sobre papel y es capturada como imagen (Plamondon, 2000), por lo que la riqueza de la información dinámica es perdida y es básicamente irrecuperable. En ambos métodos, una vez que se cuenta con la firma se procede a extraer de ella un conjunto de características que deben ser confiables, tanto para reconocer firmas genuinas como para rechazar falsificaciones, aún las del tipo bien realizadas (falsificaciones hábiles); el conjunto de características es calculado o extraído de cada una de las firmas de muestra, con lo cual se conforma un conjunto de patrones que a la vez debe servir para el entrenamiento y prueba de un clasificador. El trabajo del clasificador es aprender el comportamiento habitual de las características en una firma para posteriormente comprobar si dichas características se comportan igual en una firma de prueba.

Trabajo previo en verificación de firmas manuscritas *Off-line*. En (Justino, 2001), los autores identifican las tres clases de falsificaciones que se describen en la Tabla 1.

No.	Nombre	Descripción
1	Falsificación aleatoria	No se hace intento para reproducir el aspecto de la firma genuina. No se desea parecido entre la firma auténtica y la falsificación.
2	Falsificación simulada	La firma es copiada descuidadamente; no muy detallada o precisa. La firma falsa "tiende" a parecerse a la genuina.
3	Falsificación hábil	La firma es muy similar a la genuina.

Tabla 1. Tipos de falsificaciones acorde a Justino (Justino, 2001).

En la literatura relativa a la verificación *off-line* generalmente se han obtenido buenos desempeños (porcentaje de error de clasificación) ante falsificaciones aleatorias, como en (Justino, 2001) del orden del 0.38% o en (Drouhard, 1994) del 3%; sin embargo, ante falsificaciones simuladas y falsificaciones hábiles el porcentaje de clasificación errónea crece dramáticamente, y por sólo mencionar un caso Fang en (Fang, 1999) reporta hasta un 33.7%; sin embargo, esta situación no está relacionada con la habilidad del investigador sino con la naturaleza misma del problema de verificación *off-line*, en gran medida por la pérdida de la información dinámica. Un factor que agrava el problema de los altos índices de error ante falsificaciones hábiles a que generalmente debe prepararse un modelo de firmar de un individuo basado solamente en unas pocas muestras, generalmente de 8 a 15 (Plamondon, 2000), lo cual genera incertidumbre probabilística tanto para reconocer firmas genuinas como para rechazar falsificaciones.

Otro de los problemas principales en la verificación *off-line* es el de establecer y extraer de las firmas de muestra un conjunto de características que sea lo suficientemente repetitivo como para permitir alta capacidad de reconocimiento de especímenes genuinos de prueba y lo suficientemente discriminante como para rechazar falsificaciones. En la literatura del tema se pueden encontrar básicamente tres aproximaciones para la generación de características. Una consiste en aislar y caracterizar algunos segmentos de la firma (curvatura, suavizado) (Fang, 1999), y otra bien superponer una cuadrícula (gris) a la firma y considerar cada elemento (recuadro) de la cuadrícula como un área a ser caracterizada; un ejemplo clásico de este último se encuentra en (Sabourin, 1994) y (Sabourin, 1997), en ambos casos se busca una representación explícita, a través de vectores descriptivos de los valores de cada característica. En ocasiones se ha permitido traslapamiento de los recuadros que se consideran para representar la firma, como en (Murshed, 1995), en el cual Murshed y otros usaron cuadrícula de 16x16 píxeles con un traslapamiento del 50%. Finalmente se encuentran las aproximaciones basadas en esquemas en los que las características están implícitas en parámetros de alguna clase, como en el caso de Gouvêa en (Gouvêa, 1999), quien usó redes neuronales en su versión autoasociativa. En la última etapa, la del clasificador, se trata de decidir si la firma es genuina o no. Redes neuronales de diversos tipos se han utilizado (Gouvêa, 1999), (Drouhard, 1994), (Murshed, 1995), (Plamondon, 2000); y modelos ocultos de Markov (Justino, 2001) y otros clasificadores menos sofisticados, como el de los k-vecinos más cercanos (Sabourin, 94) y la distancia mínima con medida de Mahalanobis (Fang, 1999). Sin importar el tipo de clasificador, los resultados reportados ante falsificaciones hábiles siguen muy por debajo que los logrados bajo verificación *on-line*.

Arquitectura del sistema verificador de firmas *Off-line*. El verificador del reconocedor automático de firmas manuscritas *off-line* se muestra en la Figura 5 distinguiéndose las dos fases: la de entrenamiento (parte superior) y la de verificación (parte inferior). El objetivo de la fase de entrena-

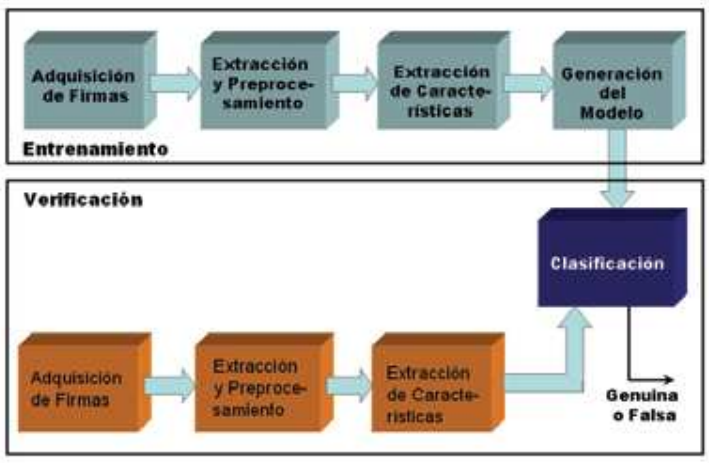

Figura 5. Arquitectura del verificador.

INVESTIGACIÓN Y CIENCIA
DE LA UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES
Número 41, (9-19), Mayo-Agosto 2008

miento es generar un modelo matemático por cada firmante, mientras que en la fase de verificación, es realizar la verificación de la firma. A continuación se describen los bloques que integran cada etapa.

B.2 Fase de entrenamiento

Los bloques de la fase de entrenamiento se encuentran en la parte superior de la Figura 5 y se describen a continuación:

Adquisición de las firmas. El primer bloque es el de adquisición de imágenes con firmas, y es efectuado con un escáner. Con el escáner se adquieren las imágenes de las firmas con una resolución de 300 dpi´s, una imagen contiene hasta 12 firmas. Las firmas son escritas en una hoja blanca de papel tamaño carta y escaneadas en escala de grises, para hacer el proceso insensible a las variaciones en el color de la tinta de la pluma.

Extracción y preprocesamiento. La imagen escaneada contiene 12 firmas igualmente espaciadas, por lo que la extracción de la firma del fondo de la hoja no representa un problema, efectuándose simplemente recortando sub-imágenes de la imagen completa. Cada sub-imagen cuenta con una firma. La Figura 6 muestra algunos ejemplos de este proceso para el firmante ficticio "John Doe".

Figura 6. Firmas extraídas del fondo de la página.

Extracción de características de la firma. Utilizando morfología matemática y con un conjunto apropiado de elementos estructurantes, es posible detectar para una firma cualquiera la posición de los trazos curvos (o equivalentemente, regiones de baja velocidad).

Una gráfica del vector característico para la firma de "John Doe" se muestra en el lado izquierdo de la Figura 7. Cada valor en dicho vector representa el número de píxeles "encendidos" (que valen 1) en la matriz que representa a la imagen erosionada, dicho número es igual al número de ocurrencias del elemento estructural

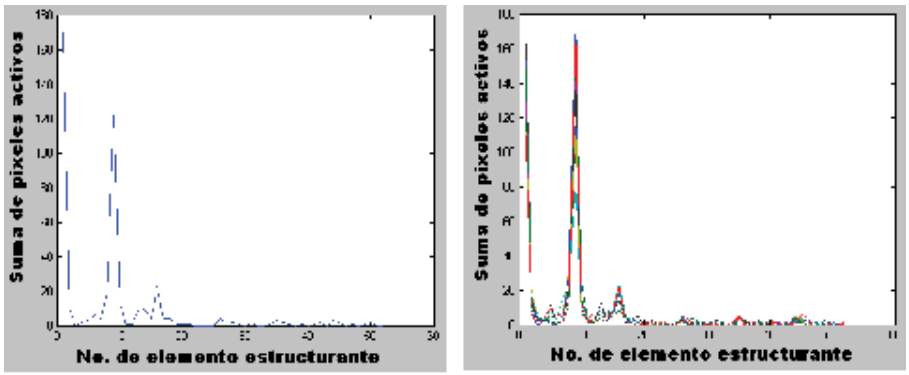

Figura 7. Vectores característicos de 10 firmas de "John Doe".

en la imagen. En el lado derecho de la Figura 7 se muestra la gráfica de 10 vectores característicos correspondientes a la misma cantidad de firmas de "John Doe". Nótese la repetibilidad en los valores entre los vectores.

Generación del modelo del firmante. Una vez binarizadas las firmas y obtenidos los respectivos vectores característicos, se generan patrones de entrenamiento para una red neuronal de retropropagación, que será el clasificador.

a) Generación de patrones de entrenamiento. En la Tabla 2 se observan los patrones de entrada y salida que le serán suministrados a la red neuronal; nótese que cada fila de la tabla es considerado como un patrón (o ejemplo de entrenamiento). La Tabla 2 se encuentra dividida en las siguientes secciones:

a.1Patrones reales: Corresponden de la fila 1 a la 10 y son formados por la erosión de cada una de las 10 firmas de un sólo firmante (para este caso, el firmante ficticio "John Doe"). Nótese que cada columna de esta sección corresponde a un valor que representa el número del elemento estructurante

Parte del Patrón	Numero de Firma	Elemento Estructurante (ocurrencias en la firma)						
		EE1	EE2	EE3	EE4	EE5	---------	EE54
Real	F1	1153	829	568	448	405	---------	393
	F2	1077	696	447	369	331	---------	227
	F3	1221	817	535	427	368	---------	393
	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-
	F10	1238	856	516	386	317	---------	276
Sintéticos Positivos	F11	1117	745	549	431	270	---------	340
	F12	1135	879	541	311	359	---------	352
	F13	1083	723	450	397	272	---------	270
	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-
	F50	1176	858	460	392	314	---------	339
Sintéticos Negativos	F51	247	252	156	213	247	---------	178
	F52	73	295	221	20	63	---------	274
	F53	80	114	10	226	70	---------	6
	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-
	F110	137	201	47	32	164	---------	226

Tabla 2: Ejemplos de entrenamiento para la red neuronal.

EEk con el cual se erosionó dicha firma, por lo tanto el número entero que aparece en la columna es la sumatoria de los bits encendidos ("1") que quedaron en la imagen digital después de erosionar la firma con el elemento estructurante en cuestión; cada renglón es un vector de características.

a.2 Patrones sintéticos positivos: Corresponden de la fila 11 a la 60; cada posición EEf,k (f, fila; k, columna) en dichas filas es calculada usando:

$$\overline{EE}_{[1\ldots10]k} - \sigma \leq EE_{f,k} \leq \overline{EE}_{[1\ldots10]k} + \sigma \qquad \text{(Ec. 1)}$$

en donde $\overline{EE}_{[1\ldots10],k}$ es el promedio de la *k*-ésima columna de los patrones reales y σ es la desviación estándar respectiva.

a.3Patrones sintéticos negativos: Corresponden de la fila 61 a la 110. Para cualquier columna de estas filas se generan números aleatorios en el rango de 1 a 300.

B.3 Clasificador de red neuronal de retropropagación

La arquitectura de la red neuronal (Bigus, 2001) de la Figura 8 está compuesta por una capa de entrada con 54 neuronas, una sola capa oculta con 108 neuronas y una neurona en la capa de salida, función sigmoidal en cada neurona.

La razón por la cual existe una neurona en la capa de salida es porque ante un ejemplo de entrenamiento suministrado a la red, ésta deberá aproximar su salida deseada a 5 (para el caso de patrones reales y sintéticos positivos) y -5 (para el caso de patrones sintéticos negativos).

Entrenamiento del clasificador. El siguiente paso es entrenar el clasificador de tal forma que sirva como herramienta de reconocimiento en la siguiente fase del verificador. El error cuadrático medio aceptable se estableció en $3x10^{-3}$, el número máximo de iteraciones -en caso de no alcanzar el error cuadrático medio- fue determinado en 1000; la razón de aprendizaje se estableció en un valor de $1x10^{-20}$.

B.4 Fase de operación, resultados de clasificación y discusión.

Cuando una firma bajo prueba es presentada al reconocedor, tienen lugar los siguientes eventos:
1. Las primeras tres etapas de la verificación se ejecutan (Figura 2), generándose los patrones respectivos de la firma.
2. Después de que los patrones son generados, la red neuronal clasificadora verifica la firma declarándola, como genuina o falsa, según se aproxime su salida a +5 ó -5.

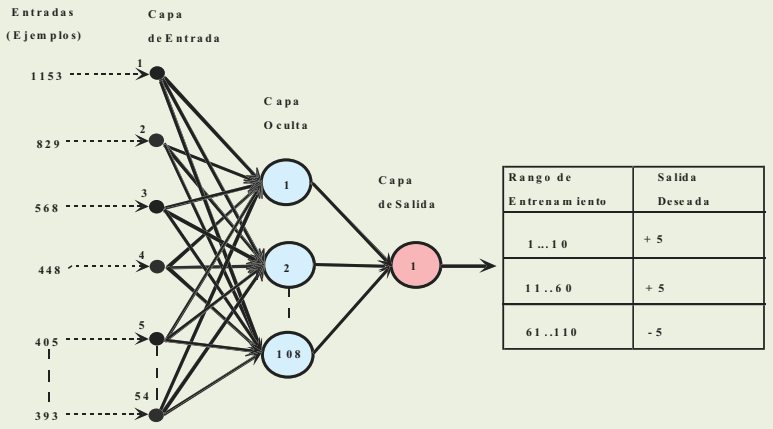

Figura 8. Arquitectura de la red neuronal.

INVESTIGACIÓN Y CIENCIA
DE LA UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES
Número 41, (9-19), Mayo-Agosto 2008

Parte del Patrón (Fase de Evaluación)	Número de Firma (Prueba)	Algunos Elementos estructurantes (ocurrencias en la firma)						Salida de la Red Neuronal
		EE1	EE2	EE3	EE4	EE5	EE6	
Real	F1	159	17	4	3	2	1	4.6868
	F2	118	6	0	3	10	5	6.6339
	F3	133	13	4	4	7	5	4.6552
	F4	182	13	2	3	7	6	4.9243
	F5	138	12	2	4	9	4	5.2071
Sintéticos Positivos	F6	134	17	2	1	-1	6	5.306
	F7	153	19	2	1	10	1	5.262
	F8	153	21	4	3	4	3	4.897
	F9	154	20	1	3	4	0	5.341
	F10	148	12	1	2	6	1	5.109
Falsificaciones (no hábiles)	F11	33	2	0	9	20	30	0.495
	F12	38	120	10	38	1	22	-0.392
	F13	51	16	120	22	21	19	-0.409
	F14	6	34	87	17	50	120	1.577
	F15	120	11	45	299	42	110	-4.582
Falsificaciones (hábiles)	F16	98	12	6	9	0	6	1.923
	F17	100	6	0	16	4	12	2.786
	F18	152	22	5	12	13	4	4.002

Tabla 3. Salidas de la red neuronal obtenidas durante la fase de operación.

La tabla 3 muestra los resultados de reconocimiento sobre el conjunto de entrenamiento y sobre firmas de prueba para una persona. Como puede verse, las firmas genuinas de prueba (F1–F5) obtienen salidas de la red neuronal bastante próximas a +5, el éxito en el reconocimiento se debe a que en el conjunto de entrenamiento se dieron a conocer a la red neuronal suficientes ejemplos (reales más sintéticos) de firmas genuinas y a que la red neuronal tiene conocimiento acerca de cómo NO deben ser dichas firmas, información contenida en el conjunto de ejemplos negativos sintéticos. Las salidas negativas y próximas a -5, tabla 3, son firmas significativamente diferentes de las genuinas, debido al conocimiento dado por los ejemplares negativos sintéticos. La red neuronal se probó adicionalmente con más ejemplares sintéticos positivos (F6–F10), los cuales fueron reconocidos como genuinos. Ante falsificaciones no hábiles (F11–F15), la red neuronal presentó buen rechazo ante falsificaciones mejor realizadas (F16– F17), pudiendo rechazar dos (de acuerdo al criterio establecido en el siguiente párrafo) y fue incapaz de rechazar una (F18).

Existe una región de incertidumbre en la salida de la red neuronal para clasificar una firma como genuina o falsa, que es la región entre +2 y +3. En términos de similitud de una firma de prueba con una firma genuina, ésta es la región en que una falsificación puede parecerse significativamente a la firma genuina; por lo tanto, la decisión debe hacerse basada en un umbral θ, siendo la efectividad del verificador afectada por este parámetro. Un θ muy bajo permitirá que firmas falsas sean clasificadas como genuinas y, en contraparte, un θ elevado ocasionará que firmas genuinas sean clasificadas como falsificaciones. La salida de la red neuronal es transformada a un grado de certidumbre en que la firma es genuina en un rango de 0-100%, para lo cual la clasificación final se da en base a una función tipo "S" parametrizada de forma tal que θ sea mapeada como se muestra en la Figura 9 mostrando además el veredicto de genuina/falsa.

RESULTADOS Y DISCUSIÓN

A. Análisis comparativo de las estrategias de balanceo de carga.

El sistema neuronal de verificación de firmas *off-line* fue implementado en la arquitectura de balanceo de carga y fue evaluado con diferentes estrategias de balanceo, cuando se evaluó con 2400 firmas incluidas en 200 imágenes con formato BMP, la estrategia propuesta mejora el tiempo de ejecución en 1.5 minutos comparada con la estrategia *Least_Loaded* y en 20.01 minutos si se compara con la otra estrategia dinámica (Umbral) como se puede observar en la Figura 10.

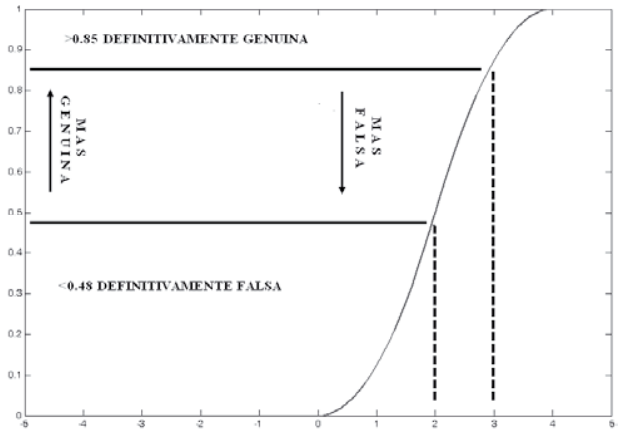

Figura 9. Toma de decisión sobre la genuinidad de una firma como función de la salida de la red neuronal.

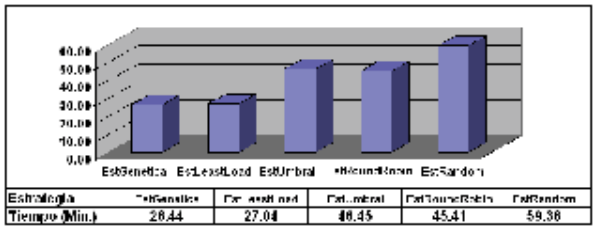

Figura 10. Tiempo de respuesta total. (Análisis comparativo de las estrategias).

También se observó cómo dicha estrategia genera tiempos de respuesta realmente considerables si se compara con las estrategias estáticas (*Round-Robin y Random*); comparada con la estrategia *Round-Robin*, la estrategia *AG + LL* mejora el tiempo de ejecución total en 18.97 minutos; comparada con la estrategia *Random*, la estrategia *AG + LL* mejora el tiempo de ejecución total en 32.92 minutos.

B. Escalando el sistema neuronal de verificación de firmas *Off-line* con la estrategia *AG + LL.*

La gráfica de la Figura 11 muestra el tiempo de respuesta total del sistema neuronal de verificación de firmas *off-line* usando la estrategia *AG + LL,* con las mismas condiciones de carga de la sección descrita previamente (2400 firmas incluidas en 200 imágenes con formato BMP), en esta figura se puede observar cómo la gráfica mantiene un comportamiento exponencial negativo en el tiempo de ejecución total cuando se escala la arquitectura de balanceo de carga de 1 a 10 réplicas.

El primer punto de la gráfica muestra cómo el tiempo de respuesta total es demasiado elevado, en este punto no se aplica ningún balanceo de carga y se utiliza sólo una réplica para procesar los requerimientos de los clientes. En el segundo punto se observa claramente cómo se reduce en 59.85% el tiempo de respuesta total cuando se escala la arquitectura a dos réplicas aplicando el balanceo de carga utilizando la estrategia *AG + LL*. En el tercer punto el tiempo de respuesta total se reduce drásticamente en 90.12% con respecto al punto inicial al escalar la arquitectura de 1 a 4 réplicas, en este punto al igual que en los puntos subsecuentes, también se aplica el balanceo dinámico de carga utilizando la estrategia *AG + LL*. En los puntos cuarto y quinto se observa que aún cuando no hay una disminución de tiempo considerable como en el punto anterior, se logra disminuir el tiempo de respuesta total en 93.09% y 94.22% respectivamente, en atención al punto inicial. Con respecto al punto tres se redujo el tiempo de respuesta en 29.99% cuando se escala la arquitectura a 8 réplicas y en 16.39% al escalarla a 10 réplicas.

C. Comportamiento del algoritmo genético para el balanceo de carga

El algoritmo genético (AG), a pesar de ser dominado por componentes altamente aleatorias, produjo resultados consistentes en cuanto al balanceo de carga, ya que en la figura 10 se mostraron valores promedio de los resultados obtenidos de ejecutar 100 veces el experimento descrito (2400 firmas), con una cantidad fija de servidores (10). Las variaciones entre ejecuciones fueron mínimas.

El tiempo de ejecución del AG no representa un *overhead* importante desde el punto de vista de la aplicación, ya que cada acción de balanceo es resuelta en un tiempo promedio aproximado de 1.6 segundos, mientras que los procesos completos sobre una firma toman 0.666 segundos, pero una corrida del AG potencialmente balancea docenas o cientos de firmas.

Para corroborar este resultado, se corrió una simulación adicional en la cual se utilizó el algoritmo de búsqueda Tabú (a 150 iteraciones) vs. el AG (a 50 generaciones, 50 cromosomas), ba-

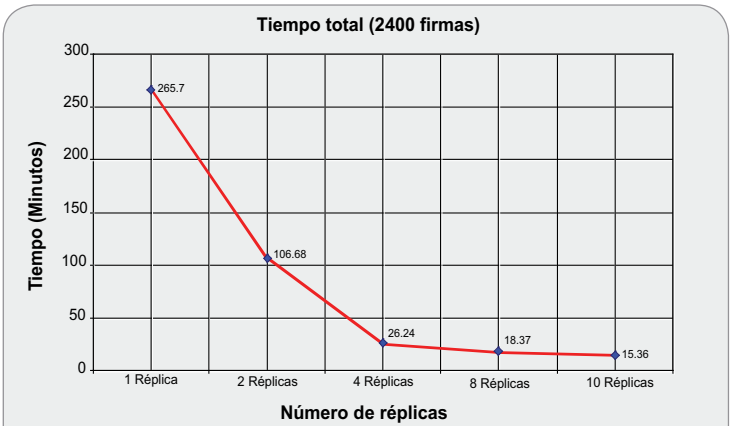

Figura 11. Tiempo de respuesta total. Balanceo dinámico de carga en un sistema de verificación de firmas manuscritas *Off-line.*

INVESTIGACIÓN Y CIENCIA
DE LA UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES
Número 41, (9-19), Mayo-Agosto 2008

lanceando 3375 requerimientos en 8 servidores, 100 corridas cada uno; en cuanto a balanceo, el AG superó a la búsqueda Tabú, pero en tiempo de ejecución el AG tardó en promedio 2.3 segundos por balanceo, mientras que la búsqueda Tabú tardó 0.57segundos. Con lo que se demuestra que la búsqueda Tabú es más rápida, pero menos efectiva en el balanceo, la razón puede ser que resuelve la optimización con mayor frecuencia encontrando mínimos locales más que globales.

CONCLUSIONES

Este artículo propuso una nueva forma de optimizar el tiempo de respuesta global utilizando una estrategia que combina un algoritmo genético (AG) con la estrategia *Leas Loaded* (LL) en un sistema neuronal de verificación de firmas fuera de línea (*off-line*). La estrategia *AG + LL* fue evaluada y comparada contra otras estrategias de balanceo dinámico de carga que son bien conocidas en la literatura, demostrando cómo la estrategia propuesta obtuvo mejores resultados en este tipo de aplicación. El sistema de verificación fue implementado en una arquitectura de balanceo dinámico de carga construida bajo el estándar de CORBA, su diseño y construcción se basó en el uso de patrones de *software*, con los cuales se construyeron modelos, que comunican la estructura y el comportamiento del sistema, para visualizar y controlar las diferentes vistas y flujos de información a un nivel de abstracción más alto que el sólo representado por las clases y los objetos, buscando con ello la futura reutilización de la arquitectura y promover el uso de buenas prácticas en el desarrollo de artefactos de *software*.

BIBLIOGRAFÍA

- BIGUS, J. *Constructing Intelligent Agents with Java, A Programmer´s Guide to Smarter Applications*, Wiley & Sons, 2001.
- BRAUN TRACY D., Siegel H. J., *et al*. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Task onto Heterogeneous Distributed Computing Systems. *Journal of Parallel and Distributed Computing*. No. 6, pp., 810-837, 2001.
- BUSCHMAN, F. *et al. A system of patterns, Pattern-Oriented, Software-Architecture*. John Wiley and Sons Ltd,1996
- BUSCHMAN F. K, H., *et al*. *Pattern Oriented Software Architecture "A pattern Language for Distributed Computing"*. Vol. 4, Viley, 2007.
- CISCO SYSTEMS Inc., High Availability Services, www.cisco.com/warp/public/cc/so/neso/ibso/s390/ mnibm_wp.htm, 2000.
- CURRY, E., CH. D., *et al*. Extending Message-Oriented Middleware Using Interception, *Proc. 3rd Int'l Workshop on Distributed Event-Based Systems (DEBS 04),* pp. 32–37, Institute of Eng. and Tech., 2004.
- DOUGLASS, S., S, M., *et al., Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects.* Vol. 2, Chichester: Wiley, 2000.
- DROUHARD, J.P., S. R., *et al*. Evaluation of a training method and of various rejection criteria for a neural network classifier used for off-line signature verification. *IEEE International Conference on Neural Networks, IEEE World Congress on Computational Intelligence*, pp., 4294-4299, NY, 1994.
- FANG, B., W. Y., *et al*. An Smoothness Index Based Approach for Off-line Signature Verification. *IEEE Proceedings of the Fifth International Conference on Document Analysis and Recognition,* pp., 785-787, ICDAR, N.Y., USA, 1999.
- GAMMA, E., H. R., *et al*. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1995.
- GOLDBERG, D. E. Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, 1989.
- GOSCINSKI, A. *Distributed Operating Systems, The Logical Design*, Addison Wesley, 1992.
- GOUVEA, R. y V. G. C. Off-line Signature Verification Using an Autoassociator Cascade-Correlation Arquitecture. *IEEE Proceedings of the Fith International Conference on Document Analysis and Recognition*, pp. 2882-2886, NY, USA, 1999.

- JUSTINO, E. J., *et al*. Off-line signature verification using HMM for random, simple and skilled forgeries. *IEEE Proceedings of the Sixth International Conference on Document Analysis and Recognition,* pp., 1031-1034, Seattle, WA, USA., 2001.

- LINDERMEIER, M. Load Management for Distributed Object-Oriented Enviroments. *In 2nd International Symposium on Distributed Objects and Applications (DOA 2000)*, Antwerp, Belguim, Sept. OMG, 2000.

- LUNA, F.J., A. R., Aplicando un Algoritmo Genético para Balancear Carga Dinámicamente en Ambientes Distribuidos Orientados a Objetos (CORBA). *Congreso Mexicano de Computación Evolutiva (COMCEV´03-CIMAT),* Guanajuato Gto, México, Mayo 28-30, 2003.

- LUNA, F.J., A. R., Combining Genetic Strategy with Least-Loaded to Improve Dynamic Load Balancing in CORBA. *The 2005 International Conference on Paralled and Distributed Processing Techniques and Applications. In Computer Science & Computer Engineering*, Las Vegas Nevada, USA, June 27-30, 2005.

- MURSHED, N., B. F., *et al*. Off-line Signature Verification, Without a Priori Knowledge of Class w2. *IEEE Proceedings of the Third International Conference on Document Analysis and Recognition*, pp. 191-196. N.Y., USA, 1995.

- OMG., Load Balancing and Monitoring for CORBA-based Applications, *Request for Proposal,* Tech. Rep., OMG Document (orbos/2001-04-27), Apr., 2001

- OSSAMA, O., O´R C., S. Strategies for CORBA Middleware-Based Load Balancing, Vol. 2, Number 3,*IEEE Distributed Systems on Line*, March, 2001.

- OSSAMA, O., O´R C., S. Designing an Adaptive CORBA Load Balancing Service Using TAO. *IEEE Distributed Systems on Line,* Vol. 2, Apr, 2001.

- PLAMONDON, R. y S.N. R., Online and off-line handwritting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:63-84, 2000.

- PUDER A., K. Roemer, MICO: An Open Source CORBA Implementation, Morgan Kaufmann, 3rd Edition, Mar., 2002.

- SABOURIN, R. y G.G., An extended-shadow-code based approach for off-line signature verification. *IEEE Proceedings of the 12th. IAPR International Conference on Computer Vision & Image Processing*, pp. 450-453. N.Y., USA. 1994.

- SABOURIN, R. y G.G., Preteux.Off-line signature verification by local granulometric size distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:976-988, 1997.

- SHIRAZI B. A., A. R. Hurson, and M. K. Kavi. *Sheduling and Load balancing in Parallel and Distributed Systems.* IEEE Computers Society Press, Los Alamitos, California USA, 1995.

- STELTING, S. y M. O., *Patrones de diseño aplicados a JAVA*, U.S.A: Pearson Educación, coedición Prentice Hall Sun Microsystems, 2003.

- ZOMAYA A. Y., Y. Hwei, Observations on Using Genetic Algorithms for Dynamic Load-Balancing, IEEE Transactions on Parallel and Distributed Systems, Vol 12. No. 9, 2001.