Balance de carga adaptativo basado en un sistema híbrido neuro-difuso para sistemas distribuidos de uso intensivo de CPU

Marcelo Cena María Liz Crespo Carlos Kavka

{mcena,mcrespo,ckavka@unsl.edu.ar}
Grupo de Interés en Sistemas Operativos *
Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950 - 5700 - San Luis
Argentina

Resumen

La performance de un sistema distribuido puede ser mejorada si se utilizan estaciones de trabajo ociosas o con poca carga, para la ejecución de tareas que son asignadas inicialmente a estaciones de trabajo que no tienen capacidad suficiente. En este trabajo se presenta una estrategia para balance de carga en sistemas distribuidos para uso intensivo de CPU, basada en un sistema híbrido neuro-difuso. Se proveen además resultados experimentales obtenidos al evaluar la estrategia en un sistema distribuido simulado.

Palabras claves: sistemas distribuidos, balance de carga, sistemas neuro-difusos.

*El grupo está auspiciado por la Universidad Nacional de San Luis, el CONICET y la Subsecretaría de Informática y Desarrollo. Su director es el MSc. Raúl Héctor Gallard

Balance de carga adaptativo basado en un sistema híbrido neuro-difuso para sistemas distribuidos de uso intensivo de CPU

Marcelo Cena María Liz Crespo Carlos Kavka

{mcena,mcrespo,ckavka@unsl.edu.ar}
Grupo de Interés en Sistemas Operativos *
Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950 - 5700 - San Luis
Argentina

Resumen

La performance de un sistema distribuido puede ser mejorada si se utilizan estaciones de trabajo ociosas o con poca carga, para la ejecución de tareas que son asignadas inicialmente a estaciones de trabajo que no tienen capacidad suficiente. En este trabajo se presenta una estrategia para balance de carga en sistemas distribuidos para uso intensivo de CPU, basada en un sistema híbrido neuro-difuso. Se proveen además resultados experimentales obtenidos al evaluar la estrategia en un sistema distribuido simulado.

Palabras claves: sistemas distribuidos, balance de carga, sistemas neuro-difusos.

1. Introducción

Un sistema distribuido está formado por un conjunto de computadoras autónomas (estaciones de trabajo) conectadas a través de una red, y equipadas con software adecuado para proveer una visión integrada del sistema [1].

Los procesos arriban a las estaciones de trabajo en forma independiente para su ejecución. Esto puede producir que algunas estaciones de trabajo estén sobrecargadas, mientras otras permanecen ociosas o con poca carga. Se puede lograr una mejora en la performance global del sistema a través de un balance de la carga, reubicando los procesos que llegan a estaciones de trabajo sobrecargadas en estaciones de trabajo ociosas o con poca carga. Si las decisiones de migraciones de procesos (reubicación) son acertadas, la mejora en performance puede ser significativa. Si, por el contrario, las decisiones son erróneas, la performance global del sistema puede disminuir.

En este trabajo se considera un sistema distribuido basado en una red de estaciones de trabajo, cada una de las cuales ejecuta un sistema operativo multitarea (tipo Unix) con compatibilidad binaria entre ellas. Cada estación de trabajo puede tener características propias: distinta velocidad de procesamiento, capacidad de disco, memoria, etc., y puede estar ejecutando procesos de cualquier tipo: editores, compiladores, servidores de archivos, etc.

El objetivo de este trabajo es permitir la ejecución de procesos de uso intensivo de CPU sobre una red de estas características, aprovechando la capacidad de estaciones de trabajo con poca carga, de forma tal de lograr alta performance global del sistema.

*El grupo está auspiciado por la Universidad Nacional de San Luis, el CONICET y la Subsecretaría de Informática y Desarrollo. Su director es el MSc. Raúl Héctor Gallard

Para cumplir con este objetivo, es necesario contar con un indicador de la carga de cada estación de trabajo. Es fundamental que el costo de obtener este indicador sea muy bajo, para no incorporar un gasto adicional. Hay una serie de indicadores que pueden ser obtenidos directamente a partir de los sistemas tipo Unix, tales como la longitud de la cola de procesos que están compitiendo por el uso de la CPU, la utilización de la CPU, utilización de discos, etc.

Una medida que ha sido considerada tradicionalmente como un indicador de la carga de un nodo es la longitud de la cola de los procesos que están compitiendo por el uso de la CPU [1][2][3]. Algunos autores [4] opinan, sin embargo, que es un estimador muy elemental y debe ser combinado con otros, tales como el uso del disco por ejemplo.

En el sistema descripto, la longitud de cola no es un estimador adecuado. La razón se puede evidenciar a través de un ejemplo. Supongamos que una estación de trabajo que actúa como servidor de archivos tiene 5 procesos en ejecución, y otra tiene 3 procesos de uso intensivo de CPU en ejecución. La selección por longitud de cola, elegiría a la segunda, aunque no es la candidata ideal para la ejecución de un nuevo proceso de uso intensivo de CPU. Es muy probable que la primera sea más adecuada, dado que los procesos que tiene en ejecución son procesos de entrada/salida y utilizan poco la CPU.

Surge naturalmente pensar en la utilización de la CPU como un estimador adecuado. De esta forma, un proceso de uso intensivo de CPU que llega a una estación de trabajo que no tiene capacidad suficiente, es enviado a aquella estación de trabajo cuya CPU tiene la menor utilización. Lamentablemente, tampoco funciona. La razon se puede evidenciar también a través de un ejemplo. Supongamos que una estación de trabajo muy lenta tiene muy pocos procesos en ejecución (por lo tanto, la utilización de su CPU es baja), y otra muy rápida tiene alta utilización de su CPU. Es muy probable, que el proceso de uso intensivo de CPU tenga una respuesta mucho más rápida en la segunda, dado que la primera es extremadamente lenta.

El algoritmo de balance de carga debe ser no centralizado y adaptativo. No centralizado por razones de seguridad y de eficiencia en términos de tráfico en la red. Adaptativo, dado que la carga de las estaciones de trabajo puede variar en el tiempo, tanto con respecto a la cantidad de procesos de uso intensivo de CPU que son generados en cada estación de trabajo, como así también con respecto al número de procesos normales (editores, etc.) que son ejecutados.

La determinación del nodo a seleccionar por el algoritmo de balance de carga es difícil. En este trabajo, se propone la utilización de un sistema híbrido neuro-difuso (de ahora en adelante SHND) para la selección de los destinos de las migraciones de procesos. Cada estación de trabajo mantiene un SHND para tomar esas decisiones en forma independiente.

2 Sistema distribuido simulado

Los modelos analíticos que permitirían el estudio de políticas de balance de carga son muy difíciles de construir. Además en ellos se suelen ignorar factores tales como, por ejemplo, los costos de comunicación. La simulación se plantea en estos casos, como una alternativa viable y adecuada. El sistema presentado se simuló en C++ bajo un sistema operativo Unix, utilizando la biblioteca de simulación de sistemas distribuidos Parasol [5].

Desde el punto de vista de la topología, el sistema distribuido consta de un conjunto de n nodos conectados a través de una red tipo bus. La velocidad de ejecución de cada nodo se puede especificar en forma individual. Cada nodo acepta la ejecución de varios procesos en forma concurrente, teniendo una disciplina preemptive round-robin. Cada nodo mantiene una lista de procesos listos para su ejecución (ready) que esperan su correspondiente time-slice. El bus de comunicación trabaja a una velocidad fija y la disciplina de uso es FCFS (primero en llegar, primero en ser atendido).

El arribo de los procesos de uso intensivo de CPU a cada nodo está controlado por una distribución poisson, con media λ especificada en forma individual y que puede variar a medida

que avanza la simulación. Esto permite simular sobrecargas transitorias y cambios en el comportamiento. El arribo de los procesos normales a cada nodo está definido en forma independiente, donde su uso de CPU está controlado también por una distribución poisson.

Desde el punto de vista del software, en cada nodo están corriendo los procesos que arriban al sistema, ya sea en forma directa o a través de migraciones, los procesos propios del nodo, y un proceso administrador. El proceso administrador se encarga de todo lo relacionado con migraciones. Se despierta cuando recibe una solicitud de migración, a la que contesta si está o no en condiciones de recibir un proceso. En caso afirmativo, inicia la recepción del proceso y comienza su ejecución en el nodo local al haberse completado la recepción. Dependiendo de la política de balance de carga implementada, el proceso administrador realiza un broadcast a intervalos regulares de los indicadores que definen la carga actual del nodo.

Cada nodo debe indicar cual es su oferta para la ejecución de procesos de uso intensivo de CPU. Para ello mantiene dos valores límites que se mantienen constantes durante la simulación: el límite inferior y el límite superior. Si el número de procesos está por encima del límite superior, se dice que el nodo está sobrecargado (overload). Si está por debajo del límite inferior, se dice que el nodo está con poca carga (underload). Si un nodo está sobrecargado al arribar un nuevo proceso de uso intensivo de CPU, se inicia una migración. Un nodo sólo aceptará migraciones si está en situación de baja carga.

Los procesos de uso intensivo de CPU son independientes, su tiempo de ejecución es de 1 segundo y su tamaño es de 64 KB. Estos valores se mantienen fijos con el fin de disminuir el número de variables a considerar en el sistema simulado. Toda la comunicación entre estos procesos y los procesos administradores se dá a traves de ports de software utilizando el bus del sistema, por lo que pueden existir demoras en la comunicación, comunes en el caso de sobrecarga del sistema.

3 Sistemas neuro-difusos

Los sistemas difusos de inferencias están basados en los conceptos de teoría de conjuntos difusos, reglas difusas y razonamiento difuso [6].

Cuando un sistema difuso tiene como entradas y salidas valores reales, implementa un mapeo no lineal de su espacio de entrada en su espacio de salida. La ventaja de estos sistemas consiste en que son capaces de tratar con valores lingüísticos. Por ejemplo, se puede describir la utilización de una estación de trabajo como alta, media o baja, dando la definición de las funciones de pertenencia respectivas.

Para definir un sistema difuso es necesario definir las reglas difusas y las funciones de pertenencia para representar los valores lingüísticos. Su elección influencia fuertemente la calidad del sistema.

Las redes neuronales artificiales son estructuras que consisten de elementos simples de procesamiento que se conectan a través de conexiones con pesos [7]. Son capaces también de aproximar funciones o realizar otras tareas aprendiendo de ejemplos. Usualmente el tiempo de aprendizaje es largo, aunque una vez entrenadas su respuesta es casi-instantánea.

Una combinación de ambas aproximaciones ofrece la posibilidad de combinar las ventajas de ambos métodos. La red neuronal puede basarse en el conocimiento definido por el sistema difuso, y refinarlo a través de un algoritmo de aprendizaje. No se requiere un largo tiempo de entrenamiento, ya que la red parte con conocimiento incorporado, y la decisión inicial de las funciones de pertenencia y reglas no se torna tan crucial, ya que pueden ser ajustadas a medida que se utiliza el sistema.

4 Algoritmo de balance basado en SHND

El éxito de un algoritmo de balance de carga se basa en su habilidad de conocer el estado de la carga de cada uno de los restantes nodos del sistema [2]. La propia naturaleza de los sistemas distribuidos hace que sea imposible que todos los nodos conozcan esta información en forma precisa. La toma de decisiones para las migraciones se debe basar entonces en estimaciones de estos valores.

Se establecen dos formas a través de las cuales los nodos pueden intercambiar información. La primera es a través de los mismos mensajes utilizados para la negociación de las migraciones, lo que no causa ningun incremento de tráfico. Sin embargo, si se utilizaran solamente estos mensajes, los nodos que no se contactan no dispondrían de información. Para solucionar este problema, se incorpora una segunda forma de intercambio de información a través de mensajes broadcast que envían las estaciones de trabajo a intervalos regulares de tiempo. Este intervalo debe ser lo suficientemente chico como para que no se pierda la actualización, y suficientemente grande para no causar un incremento significativo en el tráfico de la red.

La información que cada nodo envía a los demás consiste de 3 valores numéricos: la utilización de la CPU en el último período, el tamaño del último período y el número de procesos de uso intensivo de CPU que fueron efectivamente ejecutados en ese período.

El nodo que desea realizar una migración, debe estimar la carga de los otros nodos en base a estos tres valores, que pueden corresponder a un período anterior y por lo tanto estar desactualizados. Una vez obtenidas las estimaciones para los n-1 nodos restantes, el nodo comienza una negociación para migración con aquel cuya estimación de carga es menor.

Para la estimación se utiliza un sistema híbrido neuro-difuso que toma estos tres valores como entrada, y produce la carga estimada como salida (figura 1).

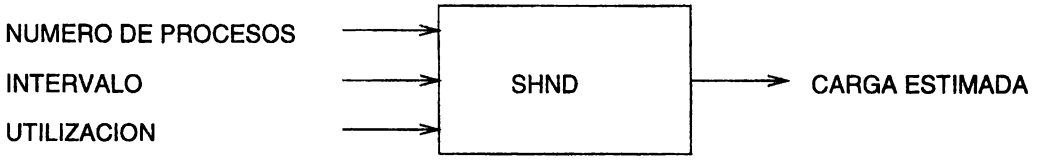

Figura 1: Sistema híbrido neuro-difuso

Al entablar la comunicación, el valor real de la carga del nodo seleccionado es transferido a través de los mensajes involucrados en la negociación. Este valor es utilizado por el nodo origen de la migración, para realizar un ciclo en el algoritmo de aprendizaje del sistema neuro-difuso. Esto ocurre tanto en el caso en que la migración sea exitosa, como si no lo es.

La carga del nodo se establece como una medida que indica que tan adecuado es el nodo para su selección para migraciones. El valor entregado por el nodo corresponde a la oferta de ejecución remota que ofrece en la situacion corriente. Un nodo que considera que sus recursos están al límite responde con un valor de carga cercano a uno. Un nodo que considera que puede aceptar más procesos responderá con un valor cercano a cero.

A continuación se presenta el algoritmo de selección del nodo destino de una migración, ejecutado por el nodo k:

```
para todo nodo i distinto de k
  predecir la carga del nodo i con el SHND
seleccionar el nodo j con menor valor de carga predecida
entablar negociacion con el nodo j
obtener la carga real del nodo j
ejecutar un ciclo de entrenamiento en el SHND
```

Cuando el nodo k recibe un mensaje de broadcast del nodo i, ejecuta el siguiente algoritmo:

```
predecir la carga del nodo i con el SHND
ejecutar un ciclo de entrenamiento en el SHND
```

5 Consideraciones sobre el algoritmo propuesto

El algoritmo presentado es adaptativo, en el sentido que, si bien parte con un conocimiento inicial sobre la predicción de cargas en el sistema, se puede adaptar, durante la ejecución, en base a su proceso de actualización a través de la ejecución de ciclos de su algoritmo de entrenamiento.

Permite que el sistema cambie dinámicamente su configuración, ya que en ningún momento se hacen suposiciones sobre el número de nodos en el sistema. Cada vez que un nuevo nodo es incorporado se registra, y puede ser tenido en cuenta como destino de migraciones.

En ningún momento se hacen suposiciones sobre la velocidad relativa de los distintos nodos. Este parámetro es determinado automáticamente por el SHND, en base al número de procesos que son ejecutados en un determinado período, teniendo en cuenta cual es el grado de utilización de la CPU.

El intercambio de información se da a través de los mismos mensajes utilizados para la negociación de la migración, por lo que no se incorpora tráfico adicional. Sólo los mensajes periódicos de actualización establecen cierto gasto extra de comunicación, el cual puede minimizarse incrementando el intervalo.

6 Descripción del SHND

El sistema propuesto se basa en la red neuro-difusa ASN (Action Selection Network) incorporada en el modelo GARIC (Generalized Approximate Reasoning-based Intelligent Control) propuesto por Berenji y Khedkar [8]. La diferencia fundamental radica en que ASN está basada en aprendizaje por refuerzo, y SHDN puede, en cambio, ser entrenada por un algoritmo de entrenamiento tradicional, ya que se conoce la salida esperada.

Se expresan los valores lingüísticos de las variables a través de funciones triangulares, definidas por tres valores: posición central (C), amplitud izquierda (I) y amplitud derecha (D). Estos valores constituyen los parámetros de la red, y serán modificados a través del algoritmo de entrenamiento.

ASN consta de cinco capas, cada una de las cuales realiza una etapa del proceso de inferencia difusa. La figura 2 muestra un ejemplo de una red ASN de dos entradas, una salida y cuatro reglas, con dos valores difusos para cada entrada y tres para la salida.

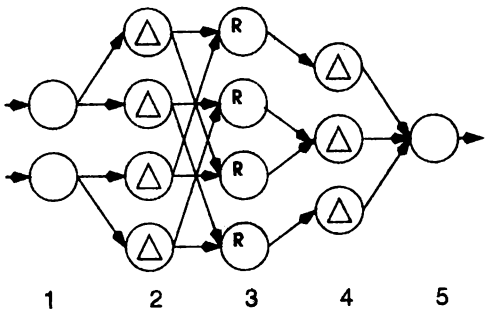

Figura 2: Ejemplo de red ASN

La capa 1 es la capa de entrada. Su función es recibir los valores reales de entrada y pasarlos a la capa 2.

Cada nodo de la capa 2 corresponde a un posible valor lingüístico de cada una de las variables de entrada, y su función es calcular la pertenencia a través de:

$$\mu(x) = \begin{cases} 1 - |x - C|/D & x \in [C, C + D] \\ 1 - |x - C|/I & x \in [C - I, C] \\ 0 & \text{en otro caso} \end{cases} \quad (1)$$

Cada nodo de la capa 3 corresponde a una regla difusa y su función es calcular la conjunción de los antecedentes. En lugar de utilizar la función mínimo usual, utiliza la función *softmin* con parámetro k [8], que es contínua y diferenciable. Obtiene como resultado w_r, que es el grado de aceptación de la regla r.

$$w_r = \frac{\sum_i \mu_i e^{-k\mu_i}}{\sum_i e^{-k\mu_i}} \quad (2)$$

Cada nodo de la capa 4 corresponde a un valor lingüístico de la variable de salida y su función es calcular el valor real usando la función de pertenencia correspondiente. Este proceso involucra la transformación de valores difusos a valores reales y se basa en el método conocido como 'media local de máximos' [8].

$$\mu^{-1}(w_r) = C + \frac{1}{2}(D - I)(1 - w_r) \quad (3)$$

El nodo de la capa 5 combina las salidas de todas las reglas en un único valor real.

$$S = \frac{\sum_r w_r \mu^{-1}(w_r)}{\sum_r w_r} \quad (4)$$

Los nodos que contienen parámetros modificables por el algoritmo de entrenamiento son los que corresponden a las capas 2 y 4 del modelo.

La modificación propuesta sobre el algoritmo de aprendizaje original no se presenta aqui debido a su extensión. Una descripción se puede encontrar en [3].

7 Definición del SHND para evaluación

Se define el SHND con tres valores lingüísticos para cada variable de entrada y cinco para la variable de salida. Esto genera un total de 27 reglas que deben ser incorporadas al sistema.

7.1 Entradas

Se definen los siguientes valores lingüísticos para cada una de las variables de entrada:

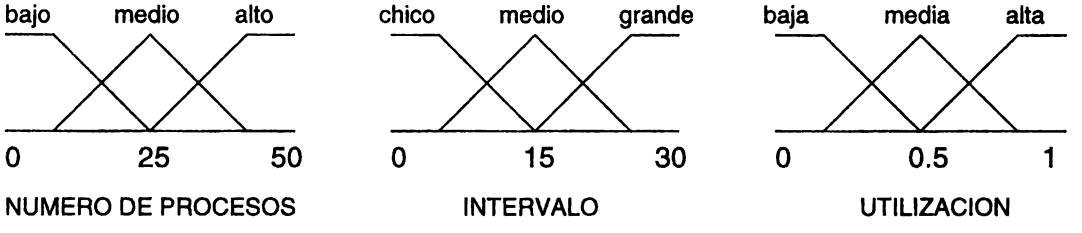

Figura 3: valores lingüísticos para las entradas

La utilización de la CPU es un valor entre 0 y 1. El número de procesos efectivamente finalizados en un período es un valor entre 0 y un máximo fijado en base a las características del sistema en 50. El tamaño del intervalo es un valor entre 0 y 30, valor que se define asumiendo que el intervalo de broadcast es de 30 segundos. Como se puede apreciar, la sobrecarga que incorporan los mensajes broadcast es mínima, teniendo en cuenta que el tiempo medio de ejecución de los procesos es de 1 segundo.

7.2 Salida

La salida del SHND se establece como un valor entre 0 y 1, que indica la carga del nodo.

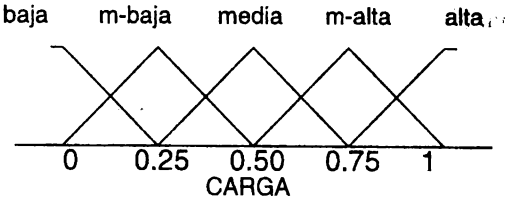

Figura 4: valores lingüísticos para la salida

7.3 Reglas

La ventaja de un SHND es que basta con que las reglas constituyan una buena aproximación inicial. Los errores serán compensados a través de entrenamiento.

Las siguientes tablas describen las reglas utilizadas en esta experimentación con el SHND. Se definen abreviaturas para las siguientes palabras: U (utilización), NP (número de procesos), TI (tamaño del intervalo), bajo (B), medio (M), alto (A), chico (C), grande (G), medio-bajo (MB) y medio-alto (MA). La primera tabla corresponde al valor medio de utilización, la segunda al valor bajo y la tercera al valor alto.

NP TI	C	M	G
B	M	MB	B
M	MA	M	MB
A	A	MA	M

NP TI	C	M	G
B	MB	B	B
M	M	MB	B
A	MA	M	MB

NP TI	C	M	G
B	MA	M	MB
M	A	MA	M
A	A	A	MA

Figura 5: reglas difusas

Las reglas son definidas teniendo en cuenta el sentido común. Por ejemplo, si el número de procesos terminados en un intervalo chico (C) es grande (A), se puede asumir que el nodo está bastante cargado (A). Si en cambio el nodo termina muy pocos (B) procesos en un intervalo grande (G), es mejor candidato para migraciones que el nodo anterior. A medida que más utilizada está su CPU disminuye la probabilidad de seleccionar ese nodo (comparando las tres tablas se puede apreciar). Esto permite tener en cuenta la velocidad de cada nodo. Un nodo más rápido generará el mismo número de procesos en un intervalo del mismo tamaño que un nodo mas lento, pero su utilización será menor, por lo que es mejor candidato.

8 Evaluación de resultados

A continuación se presentan los resultados obtenidos en cinco experimentos. En todos los casos se basaron en un sistema distribuido simulado de 10 nodos con las características descriptas en la sección 2. El tiempo de simulación fue de 3000 segundos en todos los casos. Se consideró un límite inferior de 3 y un límite superior de 4 en cada nodo. Esto significa que un nodo con tres procesos de uso intensivo de CPU en ejecución no aceptará migraciones, y un nodo con cinco procesos, intentará iniciar una migración. El intervalo de tiempo entre mensajes broadcast se fija en 30 segundos. La velocidad del enlace de comunicación se establece en 1 Mbps.

No existe, lamentablemente (en conocimiento de los autores), una política de balance de carga que pueda ser aplicada a este sistema particular. Esto hace difícil la comparación. Se

establecen, entonces, tres esquemas con los que se puede confrontar la estrategia basada en el SHND. Ellas son:

sin balance Situación en la que no se realiza ningún tipo de balance de carga. No se requiere intercambio de información.

aleatorio Las decisiones de migración son realizadas en forma aleatoria. SHND debería tener un desempeño superior a esta política para justificar su incorporación. No se requiere intercambio de información.

casi-ideal Toma las decisiones en base a los valores reales corrientes de la utilización de CPU y longitud de cola de los procesos de uso intensivo de CPU en cada nodo. Esta política no es implementable en un sistema distribuido real, ya que se requiere un conocimiento global del sistema sin demoras de transmisión. El desempeño de SHND debería acercarse al de esta política.

Los escenarios de simulación seleccionados son los siguientes:

1. Todos los nodos con igual carga.
2. Un grupo de nodos con alta carga.
3. Cambio en comportamiento.
4. Distintas velocidades de nodos.
5. Sobrecargas transitorias.

Se grafica el tiempo medio de permanencia de los procesos en el sistema con respecto a la carga del sistema. Se define la carga como el número de procesos que arriban cada 1 segundo, que es el tiempo medio de ejecución de los procesos de uso intensivo de CPU. Los gráficos incluyen sólo aquella sección del espacio de cargas, en las que se notan diferencias significativas entre los resultados de las distintas políticas, para permitir una mejor visualización.

8.1 Escenario 1: igual carga

En este escenario se asigna una carga igual a todos los nodos del sistema. El gráfico muestra como todos los algoritmos de balance de carga ofrecen una mejora en el tiempo de respuesta con respecto a una situación en la que no se realiza balance de carga en absoluto.

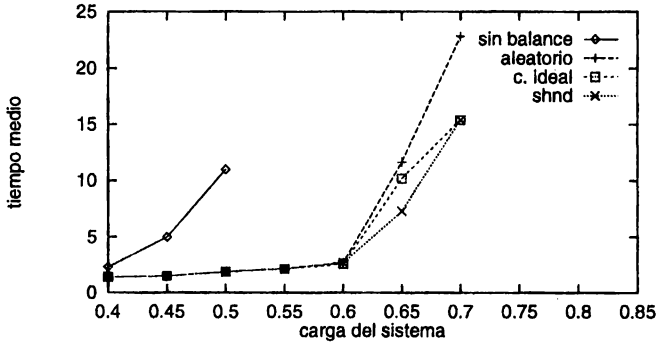

Figura 6: resultados para el escenario 1

Es importante analizar que los algoritmos de balance de carga no tienen diferencias marcadas en cuanto a performance. Esto se debe a que en cargas bajas, no se requiere hacer migraciones, y en cargas altas, los nodos no aceptarán migraciones. En todos los casos, el número de migraciones es muy bajo, y el de migraciones fallidas es muy alto.

8.2 Escenario 2: grupo con alta carga

En este escenario se define al 70% de los nodos con carga muy alta en comparación con los tres restantes. Sin algoritmo de balance de carga, se alcanza rápidamente una situación de saturación, debido a que los nodos con carga muy alta, reciben más procesos de los que son capaces de manejar.

El algoritmo aleatorio logra una mejora con respecto a la situación en la que no se hace balance, ya que distribuye la carga. Sin embargo, sólo en un 30% de los casos, el nodo seleccionado como destino de la migración estará en condiciones de aceptar el proceso. Ocurre una situación de saturación en los nodos con carga alta. Si el número de nodos con carga baja fuera mayor, lograría mucho mejores resultados, ya que en un porcentaje mayor de casos encontraría nodos en condiciones de aceptar el proceso.

El algoritmo casi ideal encuentra en todo momento un nodo capaz de satisfacer sus demandas, por lo que la carga del sistema se mantiene siempre en valores razonables.

La performance del algoritmo basado en el SHND se acerca bastante a la casi ideal.

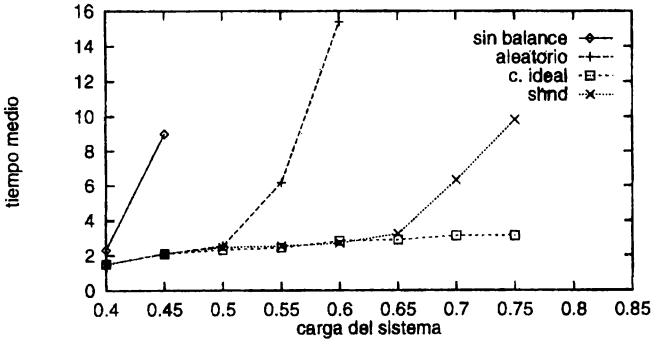

Figura 7: resultados para el escenario 2

8.3 Escenario 3: cambio en comportamiento

Esta situación es similar a la planteada en la subsección anterior. La diferencia radica en que en un momento determinado, los nodos de carga baja, pasan a tener carga alta, y el mismo porcentaje de nodos de carga alta, pasan a tener carga baja. Esto hace que los destinos de las migraciones cambien completamente.

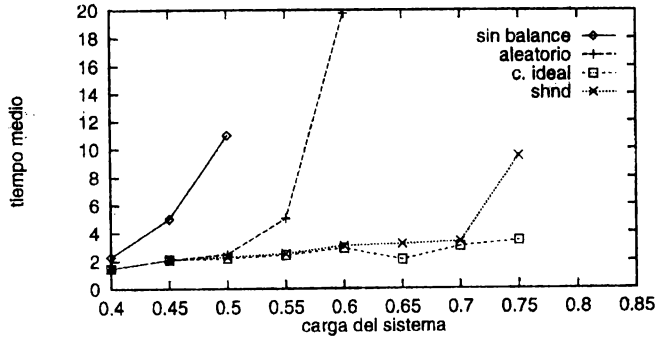

Figura 8: resultados para el escenario 3

Para las políticas aleatoria y casi ideal, el cambio no tiene efecto. La aleatoria seguirá tomando decisiones aleatorias, y la casi ideal las mejores posibles. La basada en el SHND tomará decisiones incorrectas después de la abrupta transición. Al ser rechazadas sus migraciones,

conocerá los estados de carga de los nuevos nodos cargados, por lo que no serán seleccionados nuevamente.

8.4 Escenario 4: distintas velocidades

Este escenario corresponde al caso en el que los nodos tiene distintas velocidades relativas. Se define un 50% de lo nodos lentos, y el otro 50% de nodos rápidos. Aleatoriamente se define un 70% de nodos cargados y un 30% de nodos con baja carga.

Los resultados muestran que la política basada en el SHND es aún capaz de seleccionar adecuadamente los destinos para las migraciones.

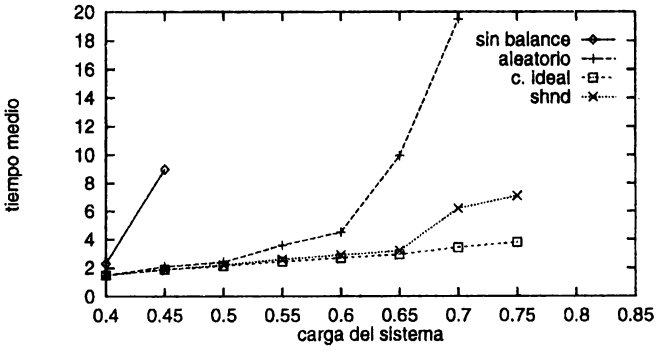

Figura 9: resultados para el escenario 4

8.5 Escenario 5: sobrecargas transitorias

Este escenario corresponde al caso en el que un 50% de los nodos recibe, durante un intervalo, una carga mayor de procesos normales. No es el mismo efecto que un incremento de la carga de procesos de uso intensivo de CPU, ya que para ellos no existen restricciones sobre su cantidad, ni se permiten migraciones.

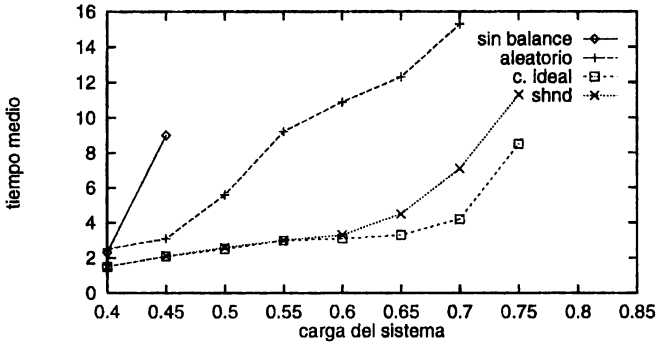

Figura 10: resultados para el escenario 5

9 Conclusiones

En este trabajo se ha presentado un algoritmo de balance de carga basado en un sistema híbrido neuro-difuso, destinado a sistemas distribuidos para uso intensivo de CPU.

La performance de un sistema distribuido es siempre superior si se incorpora algún algoritmo de balance de carga. Al igual que en otros trabajos, se ha podido apreciar que una política tan simple como la aleatoria puede ser útil, debido a su mínima necesidad de recursos del sistema.

Al experimentar con la política aqui propuesta, se ha podido comprobar que las decisiones basadas en las predicciones del SHND son muy cercanas a las ideales. Se ha afirmado [2] que el éxito de un algoritmo de balance de carga se basa en su habilidad de conocer el estado de carga de cada uno de los restantes nodos del sistema. Esta función se cumple utilizando el SHND.

En un trabajo previo, se utilizaron SHND para balance en un sistema distribuido de características diferentes. Ambos trabajos muestran que el uso de sistemas basados en lógica difusa son prometedores en este área [3].

10 Agradecimientos

Los autores agradecen a la Universidad Nacional de San Luis, al CONICET y a la Subsecretaría de Informática y Desarrollo por su auspicio. Por su importante colaboración, a todo el grupo de Investigación y a su Director.

11 Bibliografía

[1] G. Colouris, J. Dollimore and T. Kindberg. *Distributed Systems Concepts and Design.* Addison Wesley, second edition, 1994.

[2] I. D. Johnson and A. Harget. *On the Performance of Load Balancing Algorithms in Distributed Systems.* Internal Report, Aston University, Birmingham, U.K., 1992.

[3] M. Cena, M. L. Crespo y C. Kavka. *Aplicación de un sistema híbrido neuro-difuso para el balance de carga adaptativo en sistemas distribuidos.* Anales de las 25 JAIIO, Buenos Aires, 1996.

[4] M. Pankaj. *Automated Learning of Load Balancing Strategies for a Distributed Computer System.* PhD Thesis, University of Illinois at Urbana-Champaign, 1993.

[5] J. Neilson. *Parasol 3.1 User's Manual.* School of Computer Science, Carleton University, Canada, 1995.

[6] J. R. Jang and C. T. Sun. *Neuro-Fuzzy Modelling and Control.* Proceedings of the IEEE, March, 1995.

[7] L. Fausset. *Fundamental of Neural Networks, Architectures Algorithms and Applications.* Prentice Hall, Englewood Cliffs., 1994.

[8] H. R. Berenji and P. Khedkar. *Learning and Tuning Fuzzy Logic Controllers through Reinforcements.* IEEE Transactions on Neural Networks, vol. 3 no. 5, 1992.