EXPLORACIÓN EVOLUTIVA DEL EQUILIBRADO DE CARGA DINÁMICO Y DISTRIBUIDO

Mohammed Aldasht; Julio Ortega; Carlos G. Puntonet; Antonio F. Díaz
Departamento de Arquitectura y Tecnología de Computadores. Universidad de Granada

Resumen

Los algoritmos evolutivos proporcionan mecanismos para la exploración eficiente de los espacios de diseño. Así, constituyen una herramienta eficaz a la hora de identificar las mejores alternativas para implementar la solución a un problema determinado. En este trabajo aplicamos los algoritmos genéticos al ámbito del equilibrado distribuido de carga en el procesamiento paralelo. Se han clasificado y codificado las distintas estrategias de equilibrado distribuido y dinámico de carga, y se ha aplicado una metodología, basada en la computación evolutiva, capaz de analizar las características de las alternativas de equilibrado cuando se consideran distintos tipos de problemas y plataformas. Como ejemplo de aplicación de la metodología propuesta se presentan los resultados correspondientes al equilibrado dinámico de carga en un cluster de computadores para de un algoritmo paralelo de ramificación y poda.

1. Introducción

En la décimocuarta edición (año 2000) del IPDPS (*International Parallel and Distributed Processing Symposium*) se celebró una mesa redonda con el siguiente título: ¿Cuáles son los diez conceptos más influyentes del último milenio en el ámbito del procesamiento paralelo y distribuido? [1]. Uno de esos diez conceptos, entre los que están la ley de Amdahl, la segmentación de cauce, el procesamiento multihebra, la sincronización, y el procesamiento en clusters de computadores, es el equilibrado o distribución de carga (*load balancing*). El equilibrado de la carga tiene como objetivo repartir el trabajo a realizar entre los distintos nodos de procesamiento de la plataforma paralela y/o distribuida que se utiliza, de forma que se maximice la utilización de los recursos para optimizar el rendimiento de la plataforma. Los procedimientos de equilibrado de carga deben encontrar un compromiso entre la utilización de los procesadores que constituyen la plataforma y los costes asociados a la comunicación y sincronización entre esos procesadores, de manera que se minimice el tiempo de ejecución de la aplicación considerada.

El equilibrado dinámico aprovecha los recursos de comunicación de la plataforma paralela para intercambiar información de estado y tareas entre los procesadores. Por tanto, los procesadores utilizan la información local de que disponen acerca del estado global del sistema, para tomar las decisiones que permitan alcanzar los objetivos de tiempo mínimo de respuesta y rendimiento máximo. La eficiencia del procedimiento de equilibrado de carga depende del coste de comunicación entre procesadores, de la complejidad asociada al procedimiento de toma de decisiones en cada procesador, y del coste de mantener información relativa al estado global del sistema en cada uno de los nodos. Las estrategias de equilibrado de carga dinámico pueden implementarse según un modelo *centralizado* o *distribuido*. En el caso centralizado, un procesador se encarga de mantener información del estado global del sistema y, a partir de ella, lleva a cabo la asignación de tareas a procesadores. En el caso de sistemas con un número elevado de procesadores distribuidos geográficamente, esta estrategia resulta inviable, por lo que las estrategias distribuidas son las más adecuadas. En estas estrategias distribuidas, los nodos del sistema disponen de información local (más o menos actualizada) acerca del estado global del sistema y toman las decisiones de forma autónoma a partir de ese estado local y de la información que intercambian con otros procesadores. Dado que los procesadores utilizan una información parcial del estado del sistema, las decisiones que se toman no suelen corresponder a la decisión óptima.

En la literatura se describen numerosos ejemplos de este tipo de procedimientos, que

conforman un espacio de diseño bastante complejo. Consiguientemente, la toma de decisiones sobre el tipo de procedimiento a elegir, según el problema y la plataforma, resulta bastante complicada. En este artículo ilustramos las posibilidades de la computación evolutiva de los algoritmos genéticos, en el ámbito del equilibrado de carga dinámico y distribuido. Así, en la Sección 2 se describen las características de los procedimientos de equilibrado de carga distribuidos y dinámicos. La Sección 3 presenta un procedimiento genérico de equilibrado de carga al que puede aplicarse un procedimiento de optimización basado en los algoritmos genéticos. Finalmente, los resultados experimentales se muestran en la Sección 4 y las conclusiones del artículo en la Sección 5.

2. Clasificación de los procedimientos de equilibrado de carga

En los últimos años se han propuesto bastantes procedimientos de equilibrado de carga [2-6] buscando mejoras en la escalabilidad y la portabilidad. En general, se trata de nuevas propuestas para seleccionar y distribuir tareas entre procesadores que minimicen el volumen de trabajo a transferir entre procesadores, y que mantengan o mejoren la localidad en las comunicaciones del programa paralelo. Todo ello minimizando la sobrecarga asociada a la actualización y el mantenimiento de la información de carga, o bien asegurando que se encuentre dentro de límites razonables [7].

En un *procedimiento completamente distribuido*, una vez se reparten las tareas entre los procesadores, éstos se encargan tanto del procesamiento de las tareas que se les asignan, como de la redistribución de las tareas para dar respuesta a los cambios que pueden producirse en el estado de carga de los procesadores, debido a la propia dinámica de la plataforma (modificaciones del estado de carga de los distintos nodos por la presencia de otras aplicaciones), a las características de las aplicaciones (aplicaciones irregulares), o por fallos en algún procesador u otro elemento del hardware de la plataforma. En un procedimiento distribuido, todos los procesadores deben implementar, de un modo u otro los procedimientos correspondientes a las etapas de *evaluación de la carga*, *iniciación de la distribución* [5,8], *cálculo del volumen de trabajo a transferir* [5,6,9], *selección de tareas*, y *migración*. Esas etapas se implementan utilizando una serie de políticas:

- La *política de información* determina las características del intercambio de información que debe producirse entre los procesadores para que éstos puedan tener una imagen lo más actualizada posible del estado de carga de toda la plataforma. El conjunto de información local y remota que tiene un procesador es la que le permite determinar si debe iniciar una redistribución de carga, el volumen de trabajo que debe transferir, si debe actuar como emisor o receptor, etc. Las dimensiones más importantes para caracterizar una política de información están relacionadas con el marco espacial y con el marco temporal. Así, una política de información debe fijar *la topología* que definen los procesadores con información de estado mutua. Un procesador puede tener información de todos los procesadores de la plataforma, de aquellos con los que tiene relaciones de vecindad según la red de interconexión, o de grupos de procesadores que puedan establecerse con arreglo a otros criterios (por ejemplo, un grupo de procesadores seleccionados aleatoriamente) que pueden cambiar temporalmente. Además, la política de información debe establecer los instantes en los que los procesadores deben intercambiar información: política de información *periódica* o *a demanda*. En la política de información periódica, los procesadores intercambian información de su carga con una frecuencia que denominaremos frecuencia de equilibrado de carga (LBF, de *load balancing frequency*). En el caso de la política de información a demanda la información de estado se intercambia cada vez que se tiene que realizar una redistribución de la carga, ya que se supone que es el momento en que se van a producir cambios en las cargas de los procesadores que no se pueden determinar localmente.
- La *política de transferencia* establece las condiciones bajo las que debe producirse la migración de las tareas entre los procesadores. Para esto, cada procesador utilizaría la información de carga (local y remota) y establece si debe transferir tareas o solicitar que se le envíen. Así, según quién inicie las operaciones de equilibrado de carga, la política de transferencia puede ser *iniciada por el emisor*, *iniciada por el*

receptor, o *simétrica*. En el caso de la transferencia iniciada por el emisor, es el procesador que está sobrecargado y tiene que enviar tareas a otros procesadores el que comienza las operaciones de distribución de carga. Si la transferencia es iniciada por el receptor, el procesador que necesita trabajo es el que inicia las operaciones de distribución de carga. Finalmente, en la política de transferencia simétrica, el procesador que está sobrecargado y va a enviar trabajo a otro procesador y el que solicita trabajo deben sincronizarse para realizar la transferencia de uno a otro.

- La *política de localización* determina los procesadores que intervienen en las transferencias de tareas que tienen lugar en el momento de redistribuir la carga. Esta política está relacionada con la de transferencia, ya que la decisión acerca de si un procesador tiene un nivel de carga que le hace emisor o receptor corresponde, precisamente, a la política de localización. Se pueden distinguir cuatro tipos de políticas de localización: las políticas de *un umbral*, las de *dos umbrales*, las de *nivel mínimo/máximo de carga*, y las *aleatorias*. En las políticas de un umbral un nodo es emisor o receptor según su carga esté por encima o por debajo, respectivamente de un determinado nivel que se toma como umbral. En las políticas de localización de dos umbrales existen dos umbrales, uno de ellos (el umbral superior) establece el valor por encima del cual el procesador se considera sobrecargado, y el otro (el umbral inferior) representa el valor por debajo del cual es procesador tiene un nivel de carga bajo. Un procesador será emisor si su nivel de carga está por encima del umbral superior, y receptor si está por debajo del umbral inferior. A la hora de establecer los umbrales hay que tener en cuenta que la carga de la aplicación puede variar con el tiempo, por lo tanto, es conveniente utilizar umbrales que cambien dinámicamente, adaptándose al nivel de carga del sistema.
- La *política de selección* es la que determina qué tareas pueden elegirse para ser transferidas desde un procesador que ha sido designado emisor a partir de la política de localización (y en algunos casos también a través de la política de distribución, como veremos). Existen dos alternativas para esta política. Por un lado está la política de selección *con corte forzado* (*preemptive*), en la que, entre las tareas que se pueden elegir se incluye la tarea que se está procesando. En cambio, en la selección de tareas *sin corte forzado* (*no preemptive*) sólo se pueden seleccionar las tareas que están pendientes de ser procesadas. Las políticas de selección sólo necesitan información que permita estimar la carga de trabajo de cada una de ellas.
- La *política de distribución* es la que determina la forma de equilibrar el trabajo entre los procesadores que la política de localización ha seleccionado como emisores o receptores. Esta política está relacionada con las etapas de evaluación la carga de trabajo, selección de tareas, y de migración. En la política de distribución, las tareas que pueden elegirse para constituir el trabajo a intercambiar son las que permite la política de selección de tareas.

Los parámetros que deben fijarse para especificar completamente las alternativas de las distintas políticas son:

- *La granularidad, GRN*, que indica el número de tareas en que se divide el problema. Puede variar entre 1 y N, donde N es el número máximo de tareas que pueden considerarse en el problema. Un valor bajo de GRN indica una granularidad gruesa dado que se crean pocas tareas. Se utiliza un parámetro entre 0 y 1 que relaciona el valor de N y el de GRN.
- *Los umbrales, TSD1 y TSD2*, son los valores utilizados como umbrales en el caso de una política de localización con un umbral (TSD1) o con dos umbrales (TSD1 es el umbral inferior y TSD2 el umbral superior). Los valores de los umbrales están entre 1 y GRN. Se utilizan los parámetros TSDP1 y TSDP2, con valores entre 0 y 1, para establecer la relación entre TSD1 y TSD2, respectivamente, con GRN.
- *La frecuencia de equilibrado de carga, LBF*, es un parámetro que puede variar entre 1 y GRN (el parámetro LBP es un parámetro entre 0 y 1 que relaciona LBF y la granularidad). Indica que durante la ejecución del programa paralelo, un procesador puede iniciar operaciones de equilibrado de carga cada LBF intervalos de tiempo.

La clasificación de los procedimientos distribuidos de equilibrado dinámico de carga según las distintas políticas nos permite parametrizar dichos procedimientos y transformar un procedimiento genérico de equilibrado dinámico distribuido que hemos desarrollado (se

describe en la siguiente sección) en el procedimiento que se desee al fijar los parámetros a los valores correspondientes a ese procedimiento. La aproximación que seguimos es similar en muchos aspectos a la que plantea la programación genética [10], cuyo objetivo es generar programas óptimos.

```
while (not final) do
{
        if (LBC==LBF) then
        {
                distribucion_de_carga();
                LBC=0;
        }
        if (cola_trabajos no vacía) then
        {
                procesar_tarea();
                LBC=LBC+1;
                IC=IC-1;
        }
}
```

Figura 1. Llamada a la función de equilibrado

Tabla 1. Parámetros del procedimiento descrito

Param.	Defin/Rango	Significado
N	-	***Tamaño del problema.*** Estimación del número máximo de tareas del problema.
P	-	***Procesadores***
GRN	⌈GRANP×N⌉ (0<GRANP<=1)	***Granularidad.*** El problema se divide inicialmente en GRN tareas
TSD1	⌈TSDP1×GRN⌉ (0<TSDP1<1)	***Umbral inferior.***
TSD2	⌈TSDP2×GRN⌉ (0<TSDP2<1)	***Umbral superior.***
LBF	⌈LBP×GRN⌉ (0<LBP<1)	***Frecuencia de distribución de carga.***
IC		***Índice de carga.*** Número de tareas que residen en la cola de trabajos del procesador
LBC		***Contador de equilibrio de carga.***
p		Procesador actual

3. Procedimiento general de equilibrado de carga dinámico y distribuido

En esta sección se describe un *procedimiento general* de equilibrado dinámico y distribuido [12]. Las definiciones, el rango y el significado de los parámetros y variables que se utilizan en los procedimientos se proporcionan en las Tablas 1 y 2. En la Figura 1 se muestra la llamada al procedimiento de equilibrado. Como se puede ver, cada LBF intervalos de tiempo de procesamiento se realiza una llamada a la función de equilibrado, *distribución_de_carga()*. Esto no quiere decir que se tengan que iniciar forzosamente operaciones para redistribuir la carga. Cada intervalo de tiempo es igual al tiempo de procesamiento de una tarea. Por eso, cada vez que se termina una tarea se reduce el índice de carga del procesador, IC, en una unidad. Obviamente, el tamaño de la tarea, y por tanto, la duración del intervalo de tiempo dependerá de la granularidad, GRN. Como se muestra en la Tabla 1, existe una relación entre LBF y GRN que se puede controlar con el parámetro LBP, que toma valores entre 0 y 1.

Tabla 2. Parámetros del procedimiento descrito (cont.)

Parámetro	Significado
E	Número de procesadores seleccionados como emisores por la política de localización
R	Número de procesadores seleccionados como receptores por la política de localización
CE	Suma de los índices de carga (IC) de todos los procesadores emisores
CR	Suma de los índices de carga (IC) de todos los procesadores receptores
IC	Índice de carga. Número de tareas que residen en la cola de trabajos del procesador
SC	Sobrecarga. $SC = IC - \frac{CE + CR}{E + R}$
$W(T_i)$	Carga de trabajo asociada a la tarea T_i

En la Figura 2 se describen las llamadas que realiza el procedimiento de distribución de carga a las funciones que implementan las distintas políticas. Las llamadas que finalmente se realizan dependen del parámetro utilizado en la ejecución del programa según las políticas que intervienen

en la distribución de carga. Así, en la función *distribucion_de_carga()* aparecen en primer lugar tres sentencias condicionales, cada una de las cuales corresponde a una de las tres alternativas que se consideran para la política de transferencia.

```
distribucion_de_carga()
{
   if  (politica_transferencia=iniciada_por_emisor)
            {
              if (IC>TSD) then
              {
                solicita_localizar_receptor(p);
                intercambiar_información_carga
                                          (proc_receptor);
                seleccionar_tareas();
                enviar_tareas(proc_receptor);
              }
            }
    if (politica_transferencia=iniciada_por_receptor)
            {
             if (IC<TSD) then
             {
               solicita_localizar_emisor(p);
               intercambiar_información_carga
                                          (proc_emisor);
               solicitar_tareas (proc_emisor);
               recibir_tareas(proc_emisor);
             }
            }
     if (politica_transferencia=iniciada_simétricamente)
            {
              intercambiar_información_carga ();
              localizar_emisor_receptor();
              if (p=proc_emisor)
              {
                seleccionar_tareas();
                enviar_tareas(proc_receptor);
              }
              if (p=proc_receptor)
              {
                solicitar_tareas(proc_emisor);
                recibir_tareas(proc_emisor);
              }
            }
}
```

Figura 2. Procedimiento distribución de carga

Si la transferencia es iniciada por el emisor, el procesador comprueba si su índice de carga, IC, es mayor que el umbral de carga superior, TSD. Si se utiliza una política de localización con un umbral, TSD será igual a TSD1, y si se utiliza una política de localización con dos umbrales, TSD se habrá fijado igual a TSD2. Si, efectivamente, es mayor que el umbral que indica que el procesador está sobrecargado, se convierte en emisor. En este caso llama a la función *solicita_localizar_receptor()* para que le proporcione el receptor o el conjunto de receptores a los que debe enviar tareas, que se determinan posteriormente utilizando las funciones *intercambiar_informacion_carga()*, *seleccionar_tareas()*, y *enviar_tareas()*. Con respecto a la información intercambiada, aparte del índice de la carga, IC, al iniciar el sistema los nodos intercambian entre si informaciones como la velocidad del CPU, la memoria disponible, tamaño de caché, etc. Estas informaciones permiten disponer de un mejor conocimiento del estado de los procesadores, y pueden utilizarse en la localización de los procesadores adecuados para transferir carga. Además se pueden realizar los cálculos de mínimos, máximos de carga, etc. que, como veremos, se van a necesitar en algunas políticas.

En el caso de que la transferencia sea iniciada por el receptor, en primer lugar se llama a la función *solicita_localizar_emisor()*, para identificar el posible emisor del que pueda recibir las tareas que necesita el procesador. A continuación se intercambia información de carga con el emisor o emisores posibles a través de la función *intercambiar_informacion_carga()*, y se solicita el envío de tareas mediante la función *solicitar_tareas()*. Después, el procesador espera recibir estas tareas antes de proseguir con su procesamiento.

Si la transferencia se inicia simétricamente, tras intercambiar información de carga, se llama a la función *localizar_emisor_receptor()*. Esta función determina si el procesador que hace la llamada es emisor o receptor y, en cada caso, devuelve uno o varios receptores o emisores, respectivamente. Si la función *localizar_emisor_receptor()* indica que el procesador no es ni emisor ni receptor, el procesador no interviene en la redistribución de carga. Si el procesador es seleccionado como emisor llama a la función de *seleccionar_tareas()* y a la de *enviar_tareas()*, y si es designado como receptor llama a las funciones *solicitar_tareas()* y después espera recibirlas al llamar a *recibir_tareas()*.

La función *seleccionar_tareas()* permite determinar el conjunto de tareas que pueden considerarse a la hora de elegir aquellas que deben emitirse. De esta forma, en la función aparecen dos opciones que corresponden a las dos alternativas consideradas para la política de selección de tareas: con corte forzado (*preemptive*) o sin corte forzado (*no_preemptive*).

Además, en la función *seleccionar_tareas()* se llama a la función *elegir()*. Esta función es la que implementa la política de distribución, por lo que su forma depende de las características concretas del tipo de procedimiento que se considere (*difusión*, *intercambio dimensional*, etc.). La función *elegir()* se llama con un parámetro que se refiere al conjunto de tareas que pueden ser elegidas según la política de selección (con o sin corte forzado) y, en la implementación que hemos realizado, además tengan asociado un volumen de trabajo superior a un índice de carga *SC*, cuya definición aparece en la Tabla 1. En el cálculo de SC intervienen aspectos relacionados con la topología de conexión que se considere entre los procesadores (todos, los vecinos, los pertenecientes a un grupo, etc.).

La forma de esas funciones de localización es depende del tipo de política de transferencia que se utilice, poniéndose de manifiesto la interrelación entre las dos políticas. La función *localizar_emisor_receptor()*, se llama cuando la política de transferencia es simétrica. En ella aparecen las opciones correspondientes a las cuatro alternativas para la política de localización. Así según el índice de carga IC esté por debajo o por encima de TSD1 el procesador actuará como receptor o emisor, respectivamente, en el caso de que la política de localización de un umbral. En el caso de una política de localización de dos umbrales se hace lo mismo pero se utilizan los umbrales TSD1 como umbral de carga baja y TSD2 como umbral de sobrecarga. Si la política de localización se basa en la mínima/máxima carga hay que comparar el índice de carga con los valores de carga máxima y mínima del conjunto de procesadores con los que un procesador puede intercambiar carga. Estos cálculos se hacen cuando se realizan las llamadas a la función *intercambiar_información_carga()* en el caso de que se utilice esta política de localización. El conjunto de procesadores que intervienen a la hora de hacer este cálculo viene determinado por la alternativa de *topología* de la política de información. La forma de determinar el emisor y receptor en el caso de una política de localización aleatoria también precisa información de la carga máxima y mínima del correspondiente conjunto de procesadores.

Para determinar el índice (los índices) del emisor (posibles emisores) o receptor (posibles receptores), respectivamente, para el procesador designado como receptor o emisor se utilizan las funciones *emisión()* y *recepción()*. Estas funciones dan lugar a comunicación entre aquellos procesadores que pueden transferirse tareas según las alternativas topológicas de la política de distribución. Desde estas funciones, además, también se puede llamar a las funciones *localizar_emisor()* (en el caso de *emisión()*) y *localizar_receptor()* (en el caso de *recepción()*), que se describen a continuación.

La función *localizar_receptor()* permite determinar un procesador receptor en la política de transferencia iniciada por el emisor. Además de desde la función *recepción()*, la función *localizar_receptor()* se llama desde la función *solicitar_localizar_emisor()*, que genera las llamadas a *localizar_emisor(procesador)* desde *procesador* en todos aquellos procesadores que la política de distribución indique que pueden intercambiar trabajo con *procesador*. La función ejecutada en el procesador *p* envía a *solicitar_localizar_emisor()* en *procesador* el índice del procesador, *p*, si verifica de las condiciones para ser receptor que establece la política de localización que se utiliza.

La función *localizar_emisor()* es análoga a la función *localizar_receptor()* pero en este caso para el procesador o procesadores a los que se les puede solicitar tareas. Esta función se llama desde *solicitar_localizar_emisor()* en el caso de que la política de transferencia sea iniciada por el receptor (además se puede llamar desde *emisión(),* como se ha indicado).

Para explorar el espacio de diseño de los procedimientos distribuidos de equilibrado dinámico de carga se ha utilizado un algoritmo genético (Figura 3) que trabaja en el espacio definido por las distintas alternativas de las políticas de información, transferencia, localización, y selección, más los valores que pueden tomar los parámetros LBF, GRN, y los umbrales TSD1 y TSD2 (valores reales). Su objetivo es encontrar las configuraciones de las políticas y los valores de los parámetros para dichas políticas que proporcionen una mejor distribución de carga para una aplicación paralela. En la población de soluciones que evoluciona generación a generación se aplica una estrategia *elitista* dado que se mantiene la solución más idónea que se ha encontrado en la generación anterior. Precisamente, la capacidad para recordar información útil relativa a generaciones pasadas

es una de las estrategias que se ha venido considerando en la aplicación de los algoritmos evolutivos a problemas de optimización dinámica, en los que las soluciones obtenidas deben tener en cuenta la naturaleza cambiante del problema [11].

1. ***Inicializar parámetros:*** Tamaño de la Población, Probabilidades de Cruce y Mutación.
2. ***Generar una población inicial*** aleatoriamente.
3. ***Evaluar la idoneidad de los individuos de la población.*** La idoneidad de un individuo es la ganancia de velocidad que consigue el procedimiento de distribución de carga que representa.
4. ***Seleccionar una nueva población.*** La selección utiliza el método de la ruleta, pero tomando siempre el mejor individuo para la siguiente iteración.
5. ***Aplicar operadores.*** Se realiza cruce en un punto y mutación [REN96]
6. ***Ir a 3 si no se ha alcanzado la condición de final.***

Figura 3. Pseudocódigo del algoritmo genético desarrollado

4. Resultados experimentales

Los resultados que se muestran aquí corresponden a un algoritmo paralelo de ramificación y poda para el problema del viajante de comercio. En este caso el volumen de cómputo de programa no se conoce de antemano, y depende del propio proceso de búsqueda implícita que implementa el algoritmo. Así, es imprescindible utilizar un procedimiento de equilibrado de carga dinámico.

Se proporcionan (Tabla 3 y Figuras 4-6) los resultados obtenidos para un tamaño de problema correspondiente a 8 ciudades, en un cluster con 8 nodos biprocesadores a 2.4 GHz, 2 GB de memoria cada uno y Gigabit Ethernet.

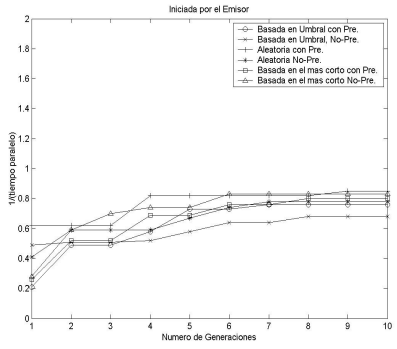

Figura 4 : Idoneidad (1/Tiempo paralelo) para las alternativas con transferencia iniciada por el emisor

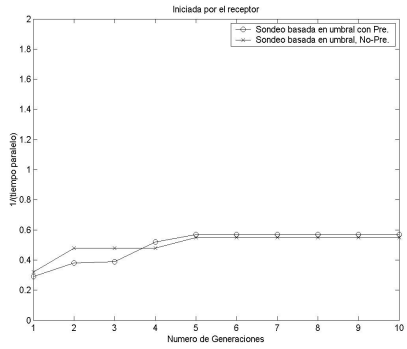

Figura 5. Idoneidad (1/Tiempo paralelo) para las alternativas con transferencia iniciada por el receptor

La Tabla 3 presenta, para los procedimientos alternativos de la política de transferencia simétrica (la que proporciona mejores valores de ganancia de velocidad), los valores de los parámetros a los que se converge el algoritmo genético y los valores de la idoneidad y de la utilización media obtenidos (entre paréntesis se indican las desviaciones típicas). Como valor de idoneidad se utiliza la inversa del tiempo de ejecución paralela (se evita evaluar el tiempo de ejecución secuencial del algoritmo, que puede llegar a ser extremadamente elevado). Las Figuras 4, 5, y 6 representan la evolución de la idoneidad a través de las sucesivas generaciones del algoritmo genético en los procedimientos alternativos de cada política de transferencia.

Los resultados obtenidos para el algoritmo de ramificación y poda muestran que la política de

transferencia simétrica proporcionan mejores valores para la ganancia de velocidad. Los procedimientos con política de transferencia iniciada por el emisor son mejores que los de transferencia iniciada por el receptor. Aquí, las utilizaciones de los procedimientos simétricos son similares en algunos casos a las de los procedimientos iniciados por el receptor, y son los procedimientos con transferencia iniciada por el emisor los que tienen utilizaciones más bajas.

En cuanto a las distintas alternativas para cada política de transferencia tenemos que en el caso de los procedimientos con transferencia iniciada por el emisor (Figura 4) y por el receptor (Figura 5) todas las alternativas convergen hacia valores bastante próximos: puede considerarse que son casi equivalentes en cuanto a la ganancia de velocidad que permiten alcanzar. En cuanto a los procedimientos con transferencia simétrica, la Figura 6 pone de manifiesto comportamientos relativamente diferentes para distintas políticas de localización y selección. En este caso, los procedimientos basados en un umbral (con o sin corte forzado) y el procedimiento basado en dos umbrales sin corte forzado son los que proporcionan mejores prestaciones.

Tabla 3 Convergencia de los parámetros para las alternativas con política de transferencia simétrica

Procedimiento	LBF	GRN	TSD1	TSD2	1/(T Par)	Util. Máx.
Umbral *(Preemption)*	107 (12)	1 (1)	101 (7)	279 (18)	1.87 (0.03)	0.09
Umbral *(No-Preemption)*	103 (8)	2 (1)	106 (8)	279 (16)	1.88 (0.02)	0.11
Dos Umbrales *(Preemption)*	111 (11)	2 (1)	109 (12)	280 (12)	1.58 (0.02)	0.17
Dos Umbrales *(No Preemption)*	116 (14)	2 (1)	94 (6)	297 (8)	1.80 (0.03)	0.13
Mín/máx *(Preemption)*	105 (7)	2 (1)	-	-	1.00 (0.01)	0.05
Mín/máx *(No Preemption)*	119 (14)	5 (3)	-	-	1.24 (0.03)	0.03

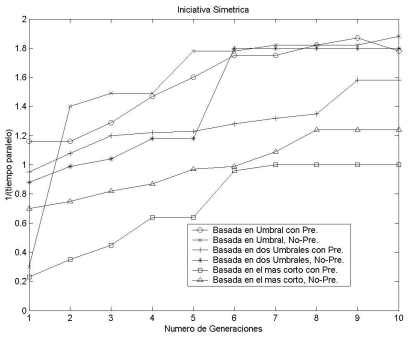

Figura 6: Idoneidad (1/Tiempo paralelo) para las alternativas con política de transferencia simétrica

La convergencia a los parámetros GRN, LBF, TSD1, y TSD2 muestra mayores desviaciones que en otros programas de prueba que se han analizado [12]. Además existe una mayor variación entre los parámetros a los que se converge según los procedimientos. Por ejemplo GRN está entre 1 y 2, aunque en algunos casos es igual a 5. Los valores de LBF a los que se converge están entre 103 y 121 para los procedimientos con transferencia iniciada por el emisor y los simétricos, pero en el caso de la transferencia iniciada por el receptor son 85 y 75, respectivamente, para las dos alternativas consideradas. Los valores de TSD1 están entre 90 y 117, y los valores de TSD2 están entre 273 y 297, aunque en un caso llega a 328 (transferencia iniciada por el emisor con localización basada en el máximo/mínimo nivel de carga sin corte forzoso). Así, el mejor procedimiento para el algoritmo de ramificación y poda aplicada al problema del viajante de comercio con 8 ciudades utiliza política de transferencia simétrica, política de localización de un umbral, y política de selección sin corte forzoso con GRN=2, LBF=103, y TSD1=106.

5. Conclusiones

A partir de las alternativas que se distinguen en las políticas de información, transferencia, localización, distribución, y selección, que definen un procedimiento de distribución de carga se han parametrizado los diferentes procedimientos y se ha aplicado la búsqueda genética en el espacio de diseño definido. Así, se puede llevar a cabo una

optimización de los parámetros que definen el comportamiento del procedimiento de equilibrado de carga, que así se aborda mediante una aproximación análoga a la que plantea la programación genética.

Los resultados experimentales obtenidos ponen de manifiesto que en todos los casos analizados la política de transferencia simétrica con política de información periódica es mejor que la iniciada por el emisor y el receptor ambas con la política de información en demanda. Estos resultados coinciden con las conclusiones obtenidas en otros trabajos [2]. En cuanto a las políticas de localización, las más eficientes suelen ser las basadas en uno o dos umbrales, y en cuanto a las políticas de selección, en algunos casos las mejores son las de corte forzoso (*preemption*) y en otros las que no utilizan corte forzoso (*no-preemption*). Los procedimientos con política de localización aleatoria son los que proporcionan las peores prestaciones.

6. Referencias

[1] Theys, M.D., et al.:"What are the Top Ten most Influential Parallel and Distributed Processing concepts of the past millenium?". Journal of Parallel and Distributed Computing, Vol.61, pp.1827-1841, 2001.

[2] Antonis, K.; Garofalakis, J.; Mourtos, I. ; Spirakis, P. : "A hierarchical adaptive distributed algorithm for load balancing". J. Parallel Distrib. Comput., 64, pp.151-162, 2004.

[3] Barker, K.; Chernikov, A.; Christoides, N.; Pingali, K.:"A Load Balancing Framework for Adaptive and Asynchronous Applications". IEEE Trans. on Parallel and Distributed Systems, Vol.15, No. 2, pp.183-192. Febrero, 2004.

[4] Watts, J; Taylor, S. "A Practical Approach to Dynamic Load Balancing". IEEE Trans. on Parallel and Distributed Systems, Vol. 9, No. 3,pp. 235-247 March 1998

[5] Xu, C.; Lau, F.:"Load Balancing in Parallel Computers". Kluwer Academic Publishers, 1997.

[6] Willebeek-LeMair, M; Reeves, A.: "Strategies for Dynamic Load Balancing on Highly Parallel Computers", IEEE Transactions on Parallel and Distributed Systems, 4:979-993, 1993.

[7] Dandamudi, S., Piotrowski, A. ,"A Comparative Study of Load Sharing on Networks of Workstations", Proceedings of the International Conference on Parallel and Distributed Computing Systems, New Orleans.

[8] Muniz, F.; Zaluska, E.:"Parallel Load-Balancing: An extension to the Gradient Model". Parallel Computing, Vol. 21, pp.287-301, 1995.

[9] Horton, G.:"A Multi-Level Difussion Method for Dynamic Load Balancing". Parallel Computing, Vol.19, pp.209-218, 1993.

[10] Koza, J.R.:"Genetic Programming: On the Programming of Computers by Means of Natural Selection". MIT Press. Cambridge, Mass., 1992.

[11] Branke, J.:"Evolutionary Optimization in Dinamic Environments". Kluwer, 2001.

[12] Aldasht, M.; Ortega, J.; Puntonet, C.G.; Díaz, A.F.:"A Genetic Exploration of Dynamic Load Balancing Algorithms". IEEE Conference on Evolutionary Computation, Portland, Oregon, 2004.