Rev. Téc. Ing. Univ. Zulia. Vol. 20, Nº 3, 203-213, 1998

Study of the task assignment problem in the distributed systems: cost functions and resolution methods

***José Lisandro Aguilar Castro*[1] *and Leila Kloul*[2]**

[1]*Dpto. de Computación, Facultad de Ingeniería, Universidad de los Andes Av. Tulio Febres Cordero 5101. Mérida, Venezuela. Email: aguilar@ing.ula.ve*

[2]*Laboratoire PRiSM, Université de Versailles St-Quentin 45, avenue des Etats-Unis, 78000 Versailles France. Email: Kloul.Leila"prism.uvsq.fr*

Abstract

In the Distributed Systems, the task allocation is one of the most important problem to take into account. This problem is NP-complete, that is why the researchers have reduced the problem dimensions deleting criteria and/or imposing constraints.

In this paper, we present the problem and we define one basic objective function, which can be used to make an optimal allocation. The definition of this function uses the following criteria: the communication cost between the processors, the task execution time, the interference cost, the load balancing cost and the reference cost to files in other sites. An adaptation of the costs to several type of architectures is presented. Finally, we present several techniques of combinatorial optimization and we apply two of them, Simulated Annealing and Genetic Algorithm, to solve the task assignment problem.

Key words: Task allocation, Distributed systems, performance evaluation, genetic algorithm, simulated annealing.

Estudio del problema de asignación de tareas en los sistemas distribuidos: funciones de costo y métodos de resolución

Resumen

En los Sistemas Distribuidos, la asignación de tareas es uno de los problemas más importantes a considerar, por esa razón ha sido objeto de numerosos estudios. Este problema es NP-completo, por lo que en la mayoría de los estudios se han simplificado las dimensiones del problema, eliminando ciertos criterios y/o imponiendo restricciones.

En este artículo, se presenta el problema en su totalidad definiendo una función de costo a partir de la cual se puede obtener una asignación óptima de las tareas. Para definir la función de costo, se ha tomado en consideración el costo de comunicación entre los procesadores, los tiempos de ejecución de las tareas, el costo debido a la interferencia entre tareas paralelas al residir en un mismo sitio, el costo por repartir la carga de trabajo y el costo debido al acceso a datos guardados en otros sitios. Se presentan ejemplos de utilización de esta función sobre distintas arquitecturas distribuidas. Finalmente, se hace un estudio de los diferentes métodos de resolución existentes y se presentan dos algoritmos para resolver el problema usando nuestra función de costo, uno basado en el Temple Simulado y el otro en los Algoritmos Genéticos.

Palabras clave: Asignación de tareas, sistemas distribuidos, evaluación de rendimiento, algoritmos genéticos, temple simulado.

Introducción

Gracias al progreso de la tecnología VLSI, al desarrollo de las Redes de Comunicación y a la necesidad de tener grandes potencias de cálculo para resolver problemas complejos, los Sistemas Distribuidos constituyen un ambiente de cómputo cada vez más usado. La modularidad y flexibilidad que caracterizan a estos sistemas, permiten adaptarlos a muchas aplicaciones en diferentes áreas, tales como control de procesos industriales, procesamiento de imágenes, etc.

La posibilidad de realizar procesamiento paralelo sobre estos ambientes, involucra la resolución de nuevos problemas antes de poder explotar esa potencialidad [1]. Uno de los principales problemas es la degradación del rendimiento. En un ambiente ideal el rendimiento debería crecer linealmente, al crecer el número de procesadores. Si uno duplica el número de procesadores en un sistema, el rendimiento debería igualmente duplicarse. Esto no sucede en la realidad, ya que a partir de un cierto número de procesadores los excesivos intercambios de mensajes de control y de transferencia de datos entre los procesadores, provocan un efecto de saturación en el sistema, lo que genera automáticamente una degradación del rendimiento (de hecho, antes de llegar a "saturación" ya existe degradación derivado de la sobrecarga en el sistema debido a la carga adicional inducida por los mensajes). Parece entonces evidente la relación entre la optimización del rendimiento y la minimización de los intercambios entre los procesadores. Existen otros aspectos que juegan un papel importante para optimizar el rendimiento, tales como el equilibrio de la carga de trabajo.

En el ciclo de vida de un programa en los Sistemas Distribuidos, su descomposición en tareas y la asignación de estas sobre los diferentes procesadores, son aspectos a considerar para optimizar el rendimiento de los sistemas [2, 3, 4]. En este artículo, solo el segundo aspecto nos interesa. Nosotros estudiamos los diferentes factores ligados al problema de asignación de tareas, para después definir una función de costo, y proponemos dos algoritmos para resolver este problema, basados en los Algoritmos Genéticos y en el Temple Simulado.

El resto del artículo esta estructurado de la siguiente manera: una definición del problema de asignación de tareas es presentada en la sección 2. En la sección 3, son definidos los diferentes criterios de optimización a considerar durante la asignación de tareas. La función de costo que proponemos es definida en la sección 4. En la sección 5, estudiamos las clásicas restricciones en estos tipos de sistemas. Ejemplos de utilización de la función de costo sobre diferentes arquitecturas distribuidas son mostrados en la sección 6. En la sección 7, presentamos un estudio de las diferentes técnicas de resolución. Además, presentamos dos algoritmos para resolver el problema de asignación de tareas, uno basado en los Algoritmos Genéticos y el otro en el Temple Simulado.

Presentación del problema

La asignación de un programa compuesto por n tareas sobre una arquitectura distribuida compuesta por P procesadores, es equivalente a la proyección $F: T \rightarrow \Pi$ según un objetivo predefinido, donde T es el conjunto de tareas y Π es el conjunto de procesadores [1, 3, 5, 6]. Existen P^n posibles asignaciones. El rendimiento del sistema depende de la asignación escogida. Para realizar esta selección varios criterios pueden ser utilizados, tales como equilibrar la carga de trabajo entre los procesadores, optimizar el grado de paralelismo, minimizar el costo de comunicación entre los procesadores, minimizar el costo de ejecución de las tareas, etc. Igualmente, la selección debe tomar en cuenta las restricciones impuestas por el programa, tales como las precedencias entre las tareas; así como las impuestas por los sistemas, tales como el tamaño de la memoria, la capacidad de los canales de comunicación, la topología de interconexión entre los procesadores, etc.

Existe una gran cantidad de trabajos dedicados al problema de asignación de tareas [1, 4, 5, 6, 7, 8, 9, 10], pero la mayoría ha reducido las dimensiones del problema eliminando ciertos criterios y/o definiendo ciertas hipótesis. De todas maneras, el problema continua siendo NP-completo [3, 5]. Por otro lado, aunque ciertos autores afirman que el deseo de minimizar los costos de comunicación entre los procesadores, de equilibrar la carga y de maximizar el grado de paralelis-

mo están en contradicción [1, 5], la definición de una función de costo como la que pretendemos realizar, permite tomar en consideración todos los criterios presentes en la resolución del problema de asignación de tareas. La asignación óptima debe buscar un buen compromiso entre estos criterios [7, 8, 11]. Las arquitecturas distribuidas donde este problema es crucial, son las arquitecturas del tipo MIMD con memoria distribuida. En estas arquitecturas, los procesadores deben comunicarse entre ellos a través de pase de mensajes.

Los diferentes criterios de asignación

La búsqueda de una asignación óptima es extremadamente compleja (NP-completo) [1, 3, 5, 6]. Esta complejidad esta ligada a las características del sistema y de los programas. Si utilizamos una función de costo, la asignación óptima será la asignación que minimice dicha función.

En esta sección, describimos los diferentes costos que intervienen en el problema de asignación de tareas. Los costos describen las características de cada uno de los elementos del sistema, así como de las tareas que componen los programas. Estos costos constituyen los criterios que deben ser optimizados por la asignación final y determinan el rendimiento del sistema.

Clásicamente, las funciones de costo han sido definidas por dos criterios: el costo de comunicación y el costo de ejecución de las tareas [1, 5, 6]. Esta definición no toma en cuenta todos los aspectos presentes a la hora de querer mejorar los rendimientos en los Sistemas Distribuidos, tales como el equilibrio de la carga entre los procesadores, etc. Durante el diseño de una función de costo, todas las características que definen a los sistemas y a los programas deben ser consideradas, sin perder de vista que esta función debe ser fácil de evaluar [3].

En el resto de esta sección, se definen los diferentes criterios tomados en cuenta en nuestra función de costo. Se supone un sistema con las siguientes características:

- No hay conexión total entre los procesadores,
- Los procesadores pueden ser heterogéneos,
- Los enlaces de comunicación son fijos,
- Las tareas y los datos pueden estar duplicados.

Costo de ejecución

Este costo depende de la complejidad de la tarea a ejecutar y de la capacidad de cálculo del procesador donde la tarea reside. El costo total de ejecución de un programa es definido, como la suma de los costos de ejecución de cada tarea del programa sobre los procesadores del sistema donde residen. Este costo es definido como:

$$C_E = \Sigma_p \Sigma_i e_i U_p X_{ip} \qquad (1)$$

donde,

e_i es la complejidad de la tarea i. En nuestro caso, lo definimos como el número de instrucciones de la tarea i. Además, suponemos el mismo grado de complejidad para todas las instrucciones.

U_p es el costo por usar el procesador p. Lo definimos como el tiempo promedio de ejecución de una instrucción en el procesador p.

X_{ip} es una variable de estado la cual es igual a 1 si la tarea i reside en el procesador p, de lo contrario es 0.

Costo de comunicación

Este costo es considerado como el criterio más importante a optimizar en un Sistema Distribuido [1, 5, 6]. El costo de comunicación depende de la cantidad de información a intercambiar entre las tareas, de la topología del sistema y de las características de los canales de comunicación. Este costo existe a partir del momento en que dos tareas que se encuentran en procesadores diferentes, desean comunicarse. Este costo es definido como:

$$C_C = \Sigma_p \Sigma_{q=p} \Sigma_i \Sigma_{j\&j\neq i} D_{pq} C_{ij} X_{ip} X_{jq} \qquad (2)$$

C_{ij} es la cantidad de información a transferir entre las tareas i y j.

D_{pq} es el costo por usar los canales de comunicación. Lo definimos como el tiempo promedio para transmitir una información, desde el procesador p hasta el procesador q.

$$D_{pq} = CCP_{piqj}X_{ip}X_{jq} + CCP_{qjpi}X_{ip}X_{jq} + r_{pqi}$$

CCP_{piqj} (respectivamente CCP_{qjpi}) es el costo por usar el procesador p (respectivamente q) para realizar la comunicación entre las tareas i y j, a sabiendas que i reside en p y j en q. Matemáticamente, puede ser expresado como:

$$CCP_{piqj} = Pr_{piqj} + AT_{pi}$$

Pr_{piqj} es el tiempo promedio que el procesador p necesita para preparar el ambiente de comunicación (negociación del protocolo de comunicación, preparación de los datos a transmitir, etc.).

At_{pi} es el tiempo promedio que debe esperar la tarea i en el procesador p, antes de poder usar los canales de comunicación. Depende del trafico en los canales de comunicación.

r_{pqi} es el tiempo promedio que se pasa en los enlaces de comunicación que conectan a p y q. Depende de la velocidad de los enlaces (canales). Este costo debe reflejar el problema de enrutamiento de la información. Es definido como:

$$r_{pqi} = \min_{s \in ch_{pq}} (\Sigma_{o,g \in s}\ 1/VT_{og} + \Sigma_{o \in s-\{p,q\}} At_{oi})$$

$$s = \{ p, ..., o, g, ..., q\},$$

donde,

ch_{pq} es el conjunto de todos los caminos posibles entre p y q.

s es el conjunto de los procesadores por donde se debe pasar para enlazar a p y q, en el camino escogido. Así, (o, g) es un canal que enlaza directamente a los sitios o y g en un camino entre p y q.

Vt_{og} es la velocidad del canal de comunicación que une directamente a dos procesadores o y g.

Los detalles del parámetro D_{pq} de la fórmula propuesta en (2) provienen del protocolo de comunicación usado, así como de la topología de interconexión del sistema. Si no se consideran esos detalles, esta función puede ser definida de una manera más simple:

$$C_C = \Sigma_p\Sigma_{q \neq p}\Sigma_i\Sigma_{j\&j \neq i}\ C_{ij}X_{ip}X_{jq}$$

Costo de referencia a un archivo

Es el costo que se origina cuando una tarea i referencia a datos contenidos en un archivo f, el cual reside en un procesador diferente a donde reside la tarea i. Este costo esta íntimamente ligado al problema de asignación de datos [3, 6]. Este costo es definido como:

$$C_F = \Sigma_p\Sigma_{q \neq p}\Sigma_i\Sigma_f\ D_{pq}V_{if}X_{ip}Y_{fq} \qquad (3)$$

V_{if} es la cantidad de datos promedio referenciados por la tarea i, del archivo f. Depende del tamaño y del tipo de organización (secuencial, etc.) del archivo.

Y_{fq} es una variable de estado igual a 1 si el archivo f reside en el procesador q, de lo contrario es 0.

Costo de interferencia

Es el costo en que se incurre cuando dos o más tareas, las cuales deben ser ejecutadas en paralelo, residen en el mismo procesador por lo que deben compartir los recursos. También es conocido como el costo por pérdida de paralelismo. Puede ser atribuido a dos factores:

— La competencia por usar los componentes del procesador: CPU, memoria, etc. (I^e)

— La competencia por usar los servicios de comunicación del procesador (I^c).

$$C_I = \Sigma_p\Sigma_i\Sigma_{j \neq i}\ (I_{pij}/2)\ X_{ip}X_{jp} \qquad (4)$$

donde, I_{pij} es el costo de interferencia entre las tareas i y j, si residen en el procesador p.

$$I_{pij} = I^e_{pij} + I^c_{pij}$$

$$I^e_{pij} = (e_i + e_j)U_p$$

$$I^c_{pij} = \Sigma_{q \neq p}\Sigma_{t \neq i\&t \neq j}\ [CCP_{piqt}C_{it} + CCP_{pjqt}C_{jt}]X_{tq}$$

Costo por el desequilibrio de la carga de trabajo

Una buena asignación debe asegurar el uso mínimo, pero a su vez equitativo, de los recursos del sistema [3, 5, 7, 11]. Como una buena asignación debe considerar los dos objetivos, debemos definir un costo que asegure una distribución

equilibrada de la carga de trabajo entre los procesadores del sistema. Este costo depende de la complejidad de las tareas a ejecutarse y de las capacidades de procesamiento de los procesadores. En nuestro caso, lo definimos como la varianza de la carga de trabajo entre los procesadores.

$$C_B = \Sigma_p \left([\Sigma_i e_i U_p X_{ip}] - [\Sigma_p \Sigma_i e_i U_p X_{ip}] / P \right)^2 / P \quad (5)$$

Definición de la Función de Costo General

Tomando en cuenta los diferentes costos definidos en la sección anterior, se define la siguiente función de costo general:

$$F = A_1 C_E + A_2 C_C + A_3 C_I + A_4 C_B + A_5 C_F \quad (6)$$

A_1, A_2, A_3, A_4 y A_5 definen las relaciones y el grado de importancia de los costos. Igualmente, los A_i permiten normalizar las unidades en la función de costo. Con esta función, podemos definir el problema de asignación óptima como:

$$MIN(F)$$

Es decir, el objetivo es minimizar el uso de los recursos, asegurando un equilibrio de la carga de trabajo entre los procesadores.

Diferentes restricciones en los Sistemas Distribuidos

Las características de un sistema imponen ciertas restricciones que son necesarias considerar para obtener una asignación realizable. Estas restricciones son de dos tipos, las que dependen de las características de la arquitectura (CPU, memoria, etc.) y las que dependen de los rendimientos deseados (plazos de tiempos, grado de paralelismo, etc.).

En esta sección, presentamos las restricciones que juzgamos más importantes a la hora de resolver el problema de asignación de tareas en los Sistemas Distribuidos. La selección de las restricciones a considerar depende de las características del sistema y del comportamiento deseado a nivel de rendimientos.

Restricciones de memoria

Esta restricción hace referencia a la capacidad de la memoria en cada procesador. La suma de los espacios de memoria (m_i) requeridos por las tareas residentes sobre el procesador *p*, debe ser inferior a la capacidad máxima de *p* (Mp).

$$\Sigma_i m_i X_{ip} \leq M_p \qquad \forall p = 1, \ldots, P$$

Plazos de tiempo

Esta restricción es utilizada en los sistemas en tiempo real, ya que el tiempo de respuesta es el criterio de rendimiento más importante. Esta restricción impone que el tiempo de ejecución de una tarea *i* sobre un procesador *p* (e_iUp), sea inferior o igual al tiempo de respuesta límite (Sp) permitido en ese procesador.

$$\Sigma_p \Sigma_i e_i U_p X_{ip} \leq Sp \qquad \forall p = 1, \ldots, P \ \& \ " i = 1, \ldots, n$$

Redundancia

Esta restricción es usada en los sistemas como un mecanismo para hacerlos tolerantes a fallas. Es basada en la idea de tener *r* ejemplares de una tarea *i* sobre *r* procesadores diferentes:

$$\Sigma_p X_{ip} = r_i \qquad \forall i = 1, \ldots, n$$

donde, r_i es el número de ejemplares que se quieren tener de la tarea *i*.

Relación Tarea-Procesador

Es una matriz lógica ($COMP_{ij}$) que define si una tarea *i* debe/puede ser asignada sobre un procesador *p*. Hay varias razones para definir si una tarea *i* debe/puede ser asignada a un procesador *p*. Por ejemplo, una tarea puede necesitar un recurso que no se encuentra en todos los procesadores

$COMP_{ij}$ es igual a 1 si la tarea *i* debe/puede ser asignada sobre el procesador *p*, de lo contrario es 0.

Restricciones de precedencia

Una relación de precedencia entre dos tareas de un programa es debido a una transmisión de control o de datos entre ellas. Esta restricción implica que una tarea no puede ser ejecutada,

antes de que terminen de ejecutarse las tareas que le preceden:

PRE_{ij} es igual a 1 si la tarea *j* debe ser ejecutada después de *i*, de lo contrario es 0.

Restricción de paralelismo

Esta restricción permite evitar la pérdida de paralelismo. Ella es definida por una matriz lógica PAR_{ij}.

PAR_{ij} es igual a 1 si las tareas *i* y *j* deben ser ejecutada en paralelo, de lo contrario es 0.

Esta restricción está ligada a la anterior en el sentido de que:

si $PRE_{ij} = 1 \Rightarrow PAR_{ij} = 0$

si $PAR_{ij} = 1 \Rightarrow PRE_{ij} = 0$

Capacidad de los canales de comunicación

La capacidad de los canales de comunicación también constituye una restricción del sistema.

La asignación de tareas depende directamente de las restricciones consideradas, ya que estas influyen sobre el espacio de soluciones. Dos tipos de problemas pueden surgir en cuanto a la escogencia de las restricciones, soluciones interesantes son eliminadas o soluciones que son poco interesantes son consideradas. Ambas situaciones desmejoran o alargan el proceso de búsqueda de la solución óptima.

Ejemplos de uso de la función de costo sobre diferentes arquitecturas distribuidas

La arquitectura de base es un sistema tipo MIMD con memoria distribuida. Otras características serán agregadas poco a poco.

Procesadores idénticos usando un bus compartido para comunicarse

Por ser homogéneos los procesadores, la ecuación 1 (C_E) no interviene en este tipo de arquitectura [5, 9]. Por el contrario, los problemas de equilibrio de la carga, costo de comunicación, costo de referencia a archivos y costo de interferencia persisten. Por otro lado, Up=1 y Dpq=1 $\forall$ p,q = 1, ..., P. La función de costos es:

$$F = C_C + C_B + C_F + C_I$$

donde,

$$C_C = \Sigma_p\Sigma_q\Sigma_i\Sigma_j\, C_{ij}X_{ip}X_{jq}$$

$$C_F = \Sigma_p\Sigma_q\Sigma_i\Sigma_f\, V_{if}X_{ip}Y_{fq}$$

$$C_B = \Sigma_p\,(\,[\Sigma_i\, e_iX_{ip}] - [\Sigma_p\Sigma_i\, e_iX_{ip}]/P\,)^2 \,/\, P$$

$$C_I = \Sigma_p\,[\Sigma_i\Sigma_{j\neq i}\;(e_i + e_j) + \Sigma_{q\neq p}\Sigma_{t\neq i\&t\neq j}[C_{it} + C_{jt}]\,X_{tq}]\,X_{ip}X_{jp}/2$$

Procesadores homogéneos sin que estén completamente interconectados

En este caso, la ecuación 1 continua sin tener sentido. C_B guarda la misma definición que en el caso anterior. En las ecuaciones 2 y 3 (C_C y C_F), Dpq expresa la topología de interconexión entre los procesadores, por consiguiente es diferente de 1. CCP_{piqj} será igual a AT_{pi} porque Pr_{piqj} es nulo $\forall$ p,q = 1, ..., P y $\forall$ i,j= 1, ..., n, al ser los procesadores idénticos. r_{pq} debe reflejar el problema de enrutamiento de la información. En la ecuación 4 (C_I), la nueva expresión de CCP_{piqj} debe ser considerada.

$$D_{pq} = CCP_{piqj} + CCP_{qjpi} + r_{pq}$$

donde, $CCP_{piqj} = AT_{pi}$

Procesadores heterogéneos sin que estén completamente interconectados

En este caso, se utiliza la función de costo definida en la sección 4, es decir, se consideran todos los costos estudiados en la sección 3. Las variables Up y Pr_{piqj} definen la heterogeneidad entre los procesadores [3, 10].

Diferentes Técnicas de Resolución

El problema de asignación de tareas, por ser NP-completo, es difícil de resolver usando métodos exactos [1, 3, 5]. Los métodos exactos permiten siempre encontrar una solución óptima, pero con tiempos de cálculo muy grandes. Para parámetros grandes de este problema (por ejemplo, para n=17 o P>2) la resolución por estos

tipos de métodos se hace imposible [1]. Uno debe contentarse con soluciones aproximativas. Así, los estudios realizados hasta ahora han sido orientados hacia estrategias que permiten obtener buenas asignaciones (soluciones subóptimas), a veces óptimas, en un tiempo de cálculo reducido. Estos métodos son conocidos como métodos heurísticos o aproximativos.

En esta sección, se presentan dos tipos de clasificación de los métodos aproximativos. La primera clasificación está inspirada en la teoría en la que se basan los métodos de resolución.

a. *Teoría de grafos:* estos métodos usan grafos para modelar el problema (los programas, los procesadores y sus enlaces, etc.) y aplican algoritmos basados en la teoría de grafos (corte mínimo-flujo máximo, etc.) para encontrar la asignación [3, 4, 5, 7, 11].

b. *Basados en análisis empíricos e intuiciones:* consisten en agrupar las tareas tomando en cuenta las restricciones del problema, hasta tener un cierto número de ellas en cada grupo [9, 10, 12]. Generalmente, estos métodos permiten tener soluciones rápidas, quizás óptimas.

c. *Basados en la programación matemática:* estos métodos formulan el problema de asignación de tareas como un problema de optimización combinatoria y lo resuelven usando técnicas de la programación matemática [1, 6, 8]. La idea consiste en definir una función de costo que constituirá la función a optimizar.

La segunda clasificación está basada en el proceso de búsqueda que sigue el algoritmo para obtener la solución. Esta clasificación está inspirada en [8].

a. *Algoritmos Iterativos:* son métodos que parten de una solución inicial completa y tratan de mejorarla. Un ejemplo de estos métodos es el Temple Simulado [5, 3, 7, 11].

b. *Algoritmos Constructivos:* son métodos que parten de soluciones iniciales parciales y buscan completarlas. Este procedimiento se detiene cuando la solución es completa [10, 12]. La mayoría de las técnicas empíricas e intuitivas son de este tipo.

En el diseño de un algoritmo para resolver el problema de asignación de tareas uno puede emplear diferentes métodos. La escogencia de un método depende esencialmente de la exactitud deseada y de la rapidez con que se quiera obtener.

Dos ejemplos de aplicación

En esta sección, se presentan dos ejemplos de aplicación de métodos heurísticos en la resolución del problema de asignación de tareas. Estos métodos son el Temple Simulado y los Algoritmos Genéticos. Se supone un sistema a bus compartido con K procesadores homogéneos. Las tareas son idénticas a nivel de tiempo de ejecución y de comunicación (es decir, cuando dos tareas que se encuentran en procesadores diferentes necesitan intercambiar información). Igualmente, el costo de acceso a archivos distante es el mismo para todas las tareas.

Temple Simulado

Este método utiliza los conceptos físicos de *temperatura* y de *energía* para representar y resolver el problema de optimización [3, 13]. La función de costo del problema constituye la energía del sistema, mientras que la temperatura es introducida para hacer una búsqueda aleatoria de la solución.

El método parte de una solución inicial completa, donde la escogencia puede ser aleatoria, e intenta mejorarla a través de cambios locales. Si la solución obtenida es mejor que la precedente, la solución es aceptada, sino es aceptada según una cierta probabilidad. La operación es repetida hasta un valor mínimo de temperatura.

Nosotros estudiamos en este método los siguientes parámetros [3]: la temperatura inicial (T_inicial), la probabilidad a partir de la cual los cambios son aceptados (Fac-accept) y el número de iteraciones para cada nivel de temperatura (L). Los mejores resultados fueron obtenidos para T_inicial = 90°, Fac_accep = 0.9 y L = 100n(K-1). Si 10% de las transformaciones son aceptadas, se ha llegado al estado de equilibrio para esa temperatura. El algoritmo general es:

— Definir una solución inicial (S)
— Definir una temperatura inicial (T_inicial)
— Definir la probabilidad de aceptación (Fac_accep)
— Calcular L=100*n*(K-1)

— Repetir hasta que el sistema se enfríe
  Repetir L veces
    Tomar un vecino S' de S
    Determinar Δ = costo(S') - costo(S)
    Si $\Delta \leq 0$
      S= S' y Accept++
    de lo contrario
      Si ($\Delta > 0$) y (exp(-Δ /T) > Fac_accep)
        S= S' y Accept++
  Si Accept $\geq$ L/10
    T = 0.93*T
  Si n < Accept < L/10
    T=0.965*T
  Si n > Accept
    Sistema frío

Algoritmos Genéticos

Este método esta basado en los principios de la biología evolutiva [3, 14]. El método sigue un proceso de evolución inteligente utilizando operadores evolutivos (mutación, cruzamiento, etc.). La idea general consiste en buscar un óptimo local partiendo de un conjunto de soluciones. Para esto, uno aplica sucesivamente los operadores evolutivos sobre las soluciones iniciales e intermediarias, de manera a generar nuevas y mejores soluciones.

En nuestro caso, se define un espacio de soluciones que esta compuesto por n individuos. Cada individuo es un vector con n elementos. Cada elemento representa una tarea del programa y tiene un valor entre 1 y P para indicar a cual procesador ella esta asignada. Para cada individuo, se calcula su costo usando la función de costo del problema. La población inicial es definida aleatoriamente. Como la población es constante, los peores individuos son reemplazados por los mejores individuos en cada generación. El procedimiento es detenido cuando se llega a un número predefinido de generaciones, o cuando la solución obtenida no puede ser mejorada. Definiremos dos individuos (vectores) para explicar los operadores genéticos utilizados:

X = 1 3 2 2 Y = 3 2 1 1.

El operador de cruzamiento permite la combinación de dos vectores. El punto que corta los dos vectores es definido aleatoriamente. Por ejemplo, para X e Y, los nuevos individuos pueden ser:

1 3 : 1 1 y 3 2 : 2 2

El operador de mutación cambia aleatoriamente una parte de un vector. Por ejemplo, para X:

1 : 2 1 : 2

Con este método estudiamos tres parámetros [3]: el número máximo de generaciones, el tamaño de la población y la probabilidad de utilizar el operador de mutación después del operador de cruzamiento (PM). Más grande es la población, más grandes serán los tiempos de ejecución; pero más pequeña es la población, más uno se puede alejar de la solución óptima. Como el operador de cruzamiento es ineficaz a partir de un momento dado (cuando comienza a generar los mismos individuos), la probabilidad PM permite decidir si el operador de mutación debe ser utilizado después del operador de cruzamiento. Más grande es PM, más grande será el tiempo de ejecución. El algoritmo general es:

— Generar la población inicial
— Repetir hasta la convergencia del sistema
  -Evaluar a cada individuo
  -Seleccionar los mejores individuos para reproducir
  -Reproducir los nuevos individuos usando los operadores evolutivos
  -Sustituir los peores individuos por los mejores nuevos individuos

Análisis de los resultados

Usamos grafos serie-paralelos para representar los programas paralelos, los cuales son definidos por el número de tareas (n) y el número máximo de sucesores por tarea (d). El número de sucesores para cada tarea es uniformemente generado según el intervalo [1, d]. La ejecución es medida en segundos. Nuestro principal objetivo consiste en observar la manera como son asignadas las tareas según diferentes funciones de costo y según las dos técnicas de asignación descritas previamente. Además, mientras sea posible las comparamos con las soluciones óptimas, las cuales fueron obtenidas usando un algoritmo de búsqueda exhaustiva sobre el espacio de soluciones. Las diferentes funciones de costo utilizadas son:

1. $F_1 = C_C + C_B$
2. $F_2 = C_C + C_B + C_E$

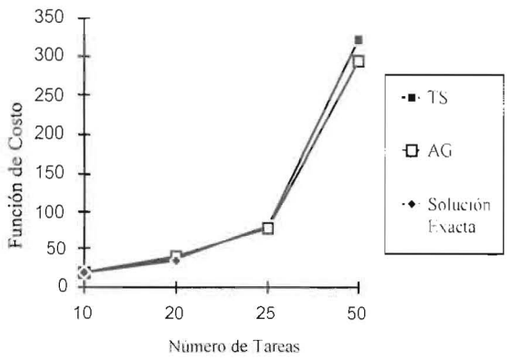

Figura 1. Optimización de la Función de Costo F1.

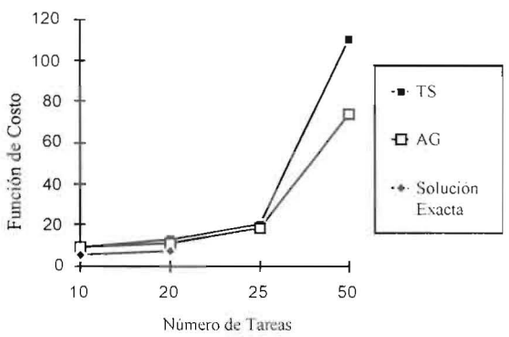

Figura 3. Optimización de la Función de Costo F5.

3. $F_3 = C_C + C_B + C_I$
4. $F_4 = C_C + C_B + C_E + C_I + C_F$
5. $F_5 = C_C + C_F$

La búsqueda de la asignación óptima para cada una de las funciones de costo, usando el Temple Simulado (TS) y los Algoritmos Genéticos (AG), nos ha permitido constatar que la escogencia del algoritmo de optimización no debe ser arbitraria, sino que depende de la función de costo a optimizar. Así, los mejores resultados para las funciones F_1, F_2 y F_3 son obtenidos con el Temple Simulado. La Figura 1 muestra los costos obtenidos con la función F_1 para programas con 50 o menos tareas y para 5 procesadores. El comportamiento de las curvas es igual para las otras dos funciones F_2 y F_3.

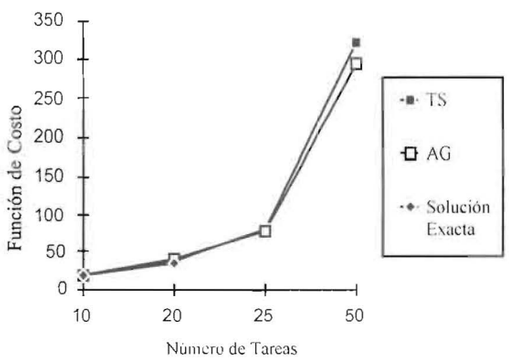

Figura 2. Optimización de la Función de Costo F4.

En cuanto a la optimización de la función de costo F_4, la escogencia del método no tiene ningún impacto sobre los resultados obtenidos (Figura 2). El comportamiento de las dos curvas es casi idéntica. Los Algoritmos Genéticos son más interesantes para la función de costo F_5, ya que la calidad de los resultados obtenidos con este método es mejor que la del Temple Simulado (Figura 3). En general, para grafos pequeños ($n > 20$ y $P > 5$) la diferencia entre las soluciones exactas y las soluciones de los métodos heurísticos no es grande. Igualmente, los tiempos de ejecución son similares.

Las asignaciones obtenidas usando las funciones de costo F_1 y F_5 son completamente diferentes (Tabla 1). Estos resultados traducen la contradicción total que existe entre los costos C_B y C_C, si consideramos que el comportamiento del costo de referencia a los archivos C_F es similar al costo de comunicación C_C.

Con las funciones de costo F_1, F_2 y F_3 han sido obtenidas asignaciones de tareas similares (Tabla 2). Por otro lado, los costos de ejecución (C_E) y de interferencia (C_I) no son contradictorios con los costos C_C y C_B. Ciertamente, C_I trata de repartir las tareas que deben ser ejecutadas en paralelo entre los procesadores del sistema, por lo que no habrá comunicación entre ellas.

Conclusiones

La asignación de tareas sobre un sistema tiene una influencia directa sobre su rendimiento. En este artículo, ha sido definida una función

Tabla 1
Asignación de tareas para n=17 y P=5, usando las funciones de costo F_1, F_5 y los algoritmos genéticos

	Procesadores				
Funciones	1	2	3	4	5
F1	7,12	6,13,15,17	3,10,11,14,16	4,9	1,2,5,8
F5	1,2,3,5,7	4	-	9,15,16,17	6,8,10,11 12,13,14

Tabla 2
Asignación de las tareas para n=17 y P=5, usando las funciones de costo F_1, F_2, F_3 y los algoritmos genéticos

	Procesadores				
Funciones	1	2	3	4	5
F1	7,12	6,13,15,17	3,10,11,14,16	4,9	1,2,5,8
F2	7,12	6,13,15,17	3,10,11,14,16	4,9	1,2,5,8
F3	7,12	6,13,15,17	3,10,11,14,16	4,9	1,2,5,8

de costo para el problema de asignación de tareas. La función de costo describe de manera detallada las características más importantes de estos sistemas, tales como la comunicación (velocidad de los enlaces, etc.), la ejecución de los procesos (costo por el uso de los procesadores, etc.). Se ha extendido el modelo de manera a incluir factores que uno no encuentra frecuentemente en la literatura, tales como la carga sobre los enlaces o la referencia a datos distantes.

Se han considerado arquitecturas donde este problema es crítico, los Sistemas de Multiprocesamiento. Igualmente, se han presentado aplicaciones de nuestra función de costo sobre diferentes arquitecturas distribuidas. Las diferentes particularidades de cada arquitectura han sido fáciles de integrar en nuestro modelo.

Este problema es NP-completo a nivel de la complejidad de resolución, por consecuencia solo los métodos aproximativos permiten obtener soluciones subóptimas, a veces óptimas, en tiempos de ejecución aceptables. Esta complejidad del problema puede ser disminuida si se agregan restricciones que caracterizan un comportamiento particular, o si se disminuye el grado de detalle del modelo (por ejemplo, no considerar la velocidad de los enlaces de comunicación). Para escoger la técnica de resolución, es necesario llegar a un compromiso entre la calidad de los resultados que se quieren y los tiempos de respuesta deseados para obtenerlos. La búsqueda de la mejor técnica para el problema de asignación de tareas es un problema abierto. Las técnicas de optimización combinatoria presentadas en este trabajo (Algoritmos Genéticos y Temple Simulado) son posibles caminos de resolución. En este trabajo, se han encontrado soluciones buenas, a veces óptimas, en tiempos de ejecución cortos.

El problema de asignación de tareas no puede ser aislado de los problemas de descomposición de programas, de asignación de archivos, y de tolerancia a fallas. La descomposición de programas define el granulado del paralelismo, y en consecuencia, las tareas a repartir en el sistema. A nivel de asignación de archivos, se ha presentado una primera manera para ligarlo al problema de asignación de tareas al introducir el costo de referencia a datos. Será interesante diseñar un modelo que considere simultáneamente los dos

problemas. La tolerancia a falla permite la continuidad en operación de los Sistemas Distribuidos, sin importar si un componente falla en un momento dado. Conceptos como migración de procesos, replica de tareas, etc. deben ser considerados. Los futuros métodos de optimización de rendimientos en los Sistemas Distribuidos deberán considerar todos estos problemas.

Referencias Bibliográficas

1. Billonnet M., Costa M. y Sutter A.: «Les problèmes de placement dans les Systèmes Distribués». Technique et Science Informatique, Vol 8, No. 4 (1989), 307-337.
2. Agrawall R. y Jaladish H.: «Partitioning techniques for Large-Grained Parallelism». IEEE Trans. on Computers, Vol 37, No. 12 (1988), 1627-1633.
3. Aguilar J.: «L'allocation de tâches, l'équilibrage de la charge et l'optimisation combinatoire». Tesis de Ph.D de la Universidad Rene Descartes, Paris, 1995.
4. Aguilar J.: «Parallel Programs: Task Graph generation and Task Allocation». Actas Científicas de la 10th International Conference on Mathematical and Computer Modelling and Scientific Computing, Boston, 1995.
5. Lo V.: «Heuristic Algorithms for Task Assignment in Distributed Systems». IEEE Trans. on Computers, Vol 37, No. 11 (1988), 1384-1397.
6. Muntean T. y Talbi G.: «Méthodes de placement statique des processus sur architectures parallèles». Technique et Science Informatique, Vol. 10, No. 5 (1991), 355-373.
7. Aguilar J.: Heuristic algorithms for Task Assignment of Parallel Programs. En: L. Dekker (ed), Massively Processing Applications and Development, Elsevier Science, Holland (1994), 146-153.
8. Andre F. y Pazat J.: «Le placement de tâches sur des architectures parallèles», Technique et Science Informatique, Vol 7, No. 4 (1988), 385-401.
9. Bollinger W. y Midkiff S.: «Heuristic Technique for Processor and Link Assignment in Multicomputers». IEEE Trans. on Computers, Vol 40, No. 3 (1991), 325-333.
10. Bowen N., Nikolaou C. y Ghafoor A.: «On the Assignment Problem of Arbitrary Process Systems to heterogeneous Distributed Computer Systems». IEEE Trans. on Computers, Vol 41, No. 3 (1992), 257-273.
11. Aguilar J.: «Maximizing the Speed-Up of Parallel Programs in Distributed Computing Environment». Actas Científicas de la IASTED International Conference on Intelligent Information Management and Systems, Washington, 1996.
12. Aguilar J. y Hernandez M.: Dynamic Task Assignment on Distributed System with Failures. En: G. Sechi (ed), Massively Parallel Computing Systems, IEEE Computer Society, USA (1996).
13. Kirkpatric S., Gelatt C. y Vecchi M.: «Optimization by Simulated Annealing», Sciences, Vol 220 (1983), 671-685.
14. Golberg D., «Genetic Algorithms in search, optimization and machine learning», Addison-Wesley, New York, 1991.

Recibido el 30 de Julio de 1996
En forma revisada el 6 de Octubre de 1997