Artículo **Revista de Sistemas Computacionales y TIC's**
Junio 2016 Vol.2 No.4 46-57

Evaluación en paralelo de los objetivos contrapuestos en la planificación y la asignación de trabajos en un sistema multiprocesadores utilizando programación multinúcleo

VELARDE, Apolinar*†

Instituto Tecnológico el Llano, México 70, 20330 Aguascalientes, Ags

Recibido Abril 28, 2016; Aceptado Junio 21, 2016

Resumen

En este trabajo se explora el problema NP-Completo de la planificación y la asignación de trabajos en un sistema de multiprocesadores. En una revisión de trabajos de investigación previos de este problema, se ha encontrado que se aborda como un problema con diferentes objetivos contrapuestos entre sí, los cuales son evaluados de forma secuencial. Por lo que en este trabajo se propone un método que permita evaluar de forma paralela dichos objetivos en diferentes núcleos de procesamiento, para acelerar la velocidad y disminuir los tiempos de las respuestas que se generaran en las asignaciones a los procesadores. El método propuesto, busca obtener una mayor rapidez de procesamiento y disminuir los tiempos de espera de los trabajos en la cola de espera.

Programación Multinucleo, algoritmo evolutivo de la Distribución Marginal Univariante, OpenMP

Abstract

In this work the NP-complete problem of planning and allocation of work in a multiprocessor system is explored. In a review of previous research work of this problem, it has been found to be approached as a problem with different conflicting objectives each other, which are evaluated sequentially. So in this paper a method to evaluate parallel those objectives, in different processing cores, to accelerate the speed and reduce the time of the responses, which were generated in allocations to processors, is proposed. The proposed method seeks to achieve faster processing and reduce waiting times, for jobs in the queue.

Multicore programming, Univariated marginal distribution algorithm, OpenMP

Citación: VELARDE, Apolinar. Evaluación en paralelo de los objetivos contrapuestos en la planificación y la asignación de trabajos en un sistema multiprocesadores utilizando programación multinúcleo. Revista de Sistemas Computacionales y TIC'S 2016, 2-4: 46-57

* Correspondencia al Autor (Correo Electrónico: avelardem@gmail.com)
† Investigador contribuyendo como primer autor.

Introducción

Las arquitecturas paralelas prevalecientes hoy en día, tales como las agrupaciones únicas y homogéneas (single homogeneous cluster), las agrupación homogéneas múltiples (múltiple homogéneos cluster) y las agrupación heterogéneas de múltiples sistemas multiprocesadores o sistemas de cómputo de alto desempeño (HPCS High Performance Computing System), se constituyen por agrupaciones (cluster por su denominación en Ingles) de procesadores concertados mediante redes de alta velocidad, que exhiben paralelismo a nivel de tarea y paralelismo a nivel de datos (P. F. Dutot, 2009). Las arquitecturas HPC soportan un modelo de memoria compartida heterogénea con acceso no uniforme a memoria (NUMA), dentro de un único nodo y memoria distribuida a través de los nodos (H. Jin, 2011). De los tipos de programación utilizados en estos sistemas, destacan la programación con el conjunto de librerías OpenMP y la programación con MPI (Message Passing Interface, de sus siglas en Ingles). OpenMP ha satisfecho, evidentemente, el campo de la informática, tanto porque es plano, y porque puede ser programado de forma incremental, preservando los programas secuenciales originales (Frias, 2006); OpenMP es un enfoque portátil para la programación paralela en los sistemas de memoria compartida, en tanto que MPI, es un estándar de facto para la programación paralela en sistemas de memoria distribuida (H. Jin, 2011).

Con el hardware paralelo y estos tipos de programación, se abordan numerosas aplicaciones que dependen de la potencia de más de un solo procesador, y cuyos resultados son críticos en el tiempo de una manera u otra (R. Chandra, 2001). Un reto importante en la programación de los sistemas multiprocesadores, es explotar el paralelismo disponible en la arquitectura y gestionar las memorias rápidas para maximizar el rendimiento (Sathe, 2011).

El problema de la planificación de tareas en sistemas multiprocesadores, es una de las aplicaciones que se busca solucionar con las arquitecturas y la programación paralelas; para solucionar este problema, se han propuesto e implementado diferentes enfoques (G. Feitelson, 2006), con los cuales se busca minimizar o maximizar las métricas de desempeño de este tipo de sistemas.

La planificación de tareas en sistemas de multiprocesadores, plantea el problema de la optimización de los recursos de computo (los núcleos de procesamiento), mediante la planificación de las tareas que permanecen en la cola de espera, y mediante la búsqueda de núcleos de procesamiento desocupados en el sistema multiprocesadores.

Este trabajo de investigación presenta, la forma en que se estructura la evaluación en paralelo de los objetivos contrapuestos de la planificación y la asignación de los trabajos en un sistema de multiprocesadores. La propuesta para evaluar en forma paralela, cada uno de los objetivos que se contraponen durante la planificación y la asignación de los trabajos que esperan por ser ejecutados en la agrupación de procesadores, surge a partir de los trabajos presentados en (referencias), en donde se realiza una evaluación secuencial de los objetivos; esta evaluación no explota el paralelismo en el procesamiento de cada objetivo, lo que provoca que los tiempos en cada métrica de desempeño sean muy cercanos y en ciertos casos iguales, a los métodos con los que se compra el método propuesto. La propuesta de paralelizar la evaluación, se refiere a procesar cada objetivo en diferente núcleo de procesamiento del nodo principal de la agrupación de procesadores.

ISSN-2444-5002

Este proceso de diseminar los procesamientos, pretende acelerar la evaluación de los objetivos y por ende comparar con mayor rapidez los resultados obtenidos de cada núcleo de procesamiento, y emitir el resultado de la mejor asignación de las tareas a los procesadores. Las experimentaciones de este trabajo, se realizan en la plataforma de desarrollo Liebres InTELigentes (Martinez, 2015), en un servidor de cuatro núcleos marca HP Proliant. Los resultados obtenidos se muestran en la sección de la evaluación del sistema.

Este trabajo, está constituido de la siguiente manera: en la sección 2, se describen los trabajos relacionados con esta investigación. La sección 3, explica la forma en que se constituye el problema de la planificación, y la asignación de las tareas paralelas en un sistema de cómputo multiprocesadores. La sección 4, describe la forma en que la investigación propuesta realiza la planificación y la asignación de las tareas en un sistema multiprocesadores, utilizando un método paralelo que busca mejorar los resultados obtenidos, con otros trabajos de investigación. La sección 5, describe detalladamente la forma en que se estructura el sistema paralelo, propuesto en este trabajo de investigación. La sección 6 experimentaciones, explica de manera detallada la forma en que se realizaron las experimentaciones,

Trabajos relacionados

Los objetivos de esta sección son dos: a) mencionar las clasificaciones que se han propuesto a través de los años de las investigaciones relacionadas con la planificación y la asignación de trabajos, y b) describir los métodos propuesto en (Apolinar, 2016) (Velarde, 2015), que representan la fundamentación para este trabajo de investigación.

El problema de la planificación y la asignación de tareas, es un problema NP-Completo considerado como uno de los más complejos de su clase por este motivo ha sido abordado con diferentes técnicas de programación, que han sido objeto de diferentes clasificaciones. Así, en (Sinnen, 2007), se especifican dos clases o categorías para la planificación de tareas, a) planificación de listas (list scheduling) y b) agrupamiento (clustering); en (Apolinar, 2016), se clasifican los trabajos relacionados dependiendo del: 1) uso que se hace del planificador y de las técnicas heurísticas empleadas, y 2) uso que se realiza del asignador, en conjunto con los modelos geométricos y las búsquedas de submallas libres a lo largo y ancho de la malla.

Los métodos heurísticos basan su funcionalidad en los algoritmos genéticos (AG). Los AG son técnicas de búsqueda global que exploran diferentes regiones del espacio de búsqueda simultáneamente, llevando un registro del conjunto de soluciones potenciales llamadas poblaciones (Goldberg, 1989). Aunque de forma alterna a los métodos heurísticos, otros métodos buscan solucionar el problema de la planificación y asignación de tareas mediante la búsqueda de procesadores libres, que se encuentren contiguos a lo largo y ancho de la malla, para lograr que las tareas asignadas permanezcan durante su ejecución, lo más cercanas posibles.

ISSN-2444-5002

En [Velarde2], se propone un método hibrido para abordar el problema de la planificación y la asignación de múltiples trabajos paralelos en un sistema multiprocesadores, como un problema multiobjetivo, que hace uso del planificador y el asignador para lograr las mejores asignaciones en la malla de procesadores que optimizan los recursos del sistema objetivo, durante el procesamiento y la finalización de las tareas. Este método utiliza un static task scheduling definida como una scheduling at compile time [19].

En [11], se aborda el problema de la planificación y la asignación de trabajos a una malla de procesadores como un problema multiobjetivo, usando para ello el algoritmo evolutivo de la Distribución Marginal Univariante (UMDA de sus siglas en Inglés, Unified Marginal Distribution Algorithm) que evalúa cada función objetivo para realizar un análisis de los resultados y determinar cuál o cuáles son los objetivos determinantes o con mayor decisión al asignar y planificar las tareas en un sistema de multicomputadoras. En este trabajo, para la realización del análisis de la contraposición de los objetivos, se utilizó el sistema de multicomputadoras Liebres Inteligentes, un sistema Sistema de Multicomputadoras para la enseñanza de la programación, de los sistemas de cómputo de alto rendimiento en instituciones de educación superior [12]. Los resultados de cada una de las funciones objetivo evaluadas, se obtienen con diferentes cargas de trabajo en la cola de espera del sistema objetivo, y con mallas de procesadores de tamaños de 4X4, 8X8 y 16X16.

Cuando los valores de las funciones objetivo son similares, se establece una escala de prioridades para el conjunto de los objetivos propuestos, de tal forma que, se realiza una aplicación sucesiva de dicha escala, hasta llegar a la mejor asignación.

Para una revisión más amplia de los trabajos en esta línea de investigación, el lector puede consultar [referencia], en donde se hace una revisión crítica de un conjunto de trabajos.

El problema de la planificación y la asignación de tareas

Esta sección define la forma en que se constituyen los trabajos en un sistema multiprocesador, después se definen los conceptos que constituyen a un sistema multiprocesador y que son usados a lo largo de este trabajo.

Composición de un trabajo

La composición de un trabajo está representada por el número de tareas establecidas por el usuario. Un trabajo de una sola tarea se define como una aplicación monolítica, en la que se especifica una sola tarea; los trabajos paralelos (o multi-tarea) están representados como un grafo dirigido a cíclico (DAG), en el cual los nodos representan las tareas particulares particionadas de una aplicación, y los bordes representan la comunicación entre las tareas; las tareas pueden ser dependientes o independientes (H. Hussain, 2013).

Las tareas independientes pueden ser ejecutadas simultáneamente para minimizar el tiempo de procesamiento; las tareas dependientes deben ser procesadas de una manera predefinida para asegurar que todas las dependencias se satisfacen.

Planificar y asignar tareas en un sistema de multiprocesadores

Las siguientes definiciones establecen la planificación y la asignación de cada uno de los trabajos y sus tareas particulares, en un sistema de multiprocesadores.

Definición 1. Dado un DAG que representa al trabajo J_n, constituido por un conjunto de tareas particulares particionadas de una aplicación: T_1, T_2, … T_n, entonces J_1,T_5 representa al trabajo 1 con 5 tareas particulares particionadas de una aplicación.

Definición 2. Dado el vector *Q* que representa la cola de trabajos en espera, conteniendo *n* trabajos para su procesamiento en el sistema de multiprocesadores, entonces Q[0] = J_1,T_5 representa la posición 0 del vector que contiene al trabajo J_1 con 5 tareas particulares particionadas de una aplicación.

Definición 3. Una malla n-dimensional tiene k_0 x k_1 x... x k_{n-2} x k_{n-1} nodos, donde k_i es el número de nodos a lo largo de la i-ésima dimensión y $k_i \geq 2$. Cada nodo se identifica por *n* coordenadas:

$$\rho_0(a), \rho_1(a),..., \rho_{n-2}(a), \rho_{n-1}(a),$$

Donde:

$$0 \leq \rho_i(a) < k_i \text{ para } 0 \leq i < n.$$

Dos nodos *a* y *b* son vecinos si y solo si $\rho_i(a) = \rho_i(b)$ para todas las dimensiones excepto para una dimensión *j*, donde $\rho_j(b) = \rho_j(a) \pm 1$.

Cada nodo en una malla se refiere a un procesador y dos vecinos están conectados por un enlace de comunicación directo.

Definición 4. Una malla 2D, la cual es referenciada como $M(W, L)$ consiste de $W \times L$ procesadores, donde W es el ancho de la malla y L es la altura de la malla. Cada procesador se denota por un par de coordenadas (x, y), donde:

$$0 \leq x < W \text{ y } 0 \leq y < L$$

Un procesador está conectado por un enlace de comunicación bidireccional a cada uno de sus vecinos. Para cada malla 2D $a=P_{ij}$.
Definición 5. En una malla 2D, $M(W, L)$, una sub-malla: $S(w, l)$ es una malla de dos dimensiones que pertenece a $M(W, L)$ con un ancho w y una altura l, donde $0 < w \leq W$ y $0 < l \leq L$. $S(w, l)$ están representadas por las coordenadas (x, y, x', y'), donde (x, y) es la esquina inferior izquierda de la sub-malla y (x', y') es la esquina superior derecha. El nodo de la esquina inferior izquierda es llamado el nodo base de la sub-malla y la esquina superior derecha es el nodo final. En este caso $w=x'-x+1$ y $l=y'-y+1$. El tamaño de $S(w, l)$ es: $w \times l$ procesadores.

Definición 6. En una malla $2D\ M(W, L)$, una sub-malla disponible $S(w, l)$ es una sub-malla que satisface las condiciones: $w \geq \alpha$ y $w \geq \beta$ asumiendo que la asignación de S (α, β) requerida, donde la asignación se refiere a seleccionar un conjunto de procesadores para una tarea de llegada.

Definición 7. Sea ϑ un conjunto de tareas del sistema, tal que $\vartheta = J_1, J_2, \ldots, J_n$ donde n es el número de tareas en el tiempo t y ϑ_k un conjunto de sub-tareas de la tarea k donde: $\vartheta_k = j_{k1}, j_{k2}, \ldots, j_{kf(k)}$ y $f(k)$ es el número total de sub-tareas de la tarea j. Para cada tarea j y cada sub-tarea $f(k) \in j$ se tiene un procesador $m_i \in P$ en el que se debe ejecutar cada tarea j y cada sub-tarea $j_{kf(k)}$, consumiendo un tiempo $t \in \mathbb{N}$ ininterrumpido.

ISSN-2444-5002

Definición 8. Dadas dos matrices de tamaño *n x n*: una matriz de flujo *F* cuyos *(i,j)* ésimos elementos representan los flujos entre tareas *i* y *j* y un arreglo de distancias *D* cuyos *(i,j)* ésimos elementos representan la distancia entre sitios *i* y *j*. Una asignación se representa por el vector *p*, el cual es una permutación de los números 1, 2,…, n. *p(j)* es el lugar donde la tarea *j* es asignada. Así, la asignación cuadrática de tareas puede ser escrita como:

$$\min_p e \sum_{i=1}^{n} \sum_{i=1}^{n} f_{ij} dp(i) p(j) \quad (1)$$

Definición 9: La correspondencia de un trabajo o tarea a un procesador libre en la malla M se define como: si ϑ es un conjunto de trabajos, and $\vartheta = J_1, J_2,\ldots, J_n$ donde *n* es el número de trabajos en el tiempo *t* y ϑ_k es un conjunto de tareas del trabajo *k* donde: $\vartheta_k = j_{k1}, j_{k2},\ldots, j_{kf(k)}$ y *f(k)* es el total de tareas del trabajo *j*, entonces para cada *j* y cada tarea *f(k)* $\in j$ existe un procesador *(x, y)* $\in M$, en el cual se ejecuta cada trabajo *j* o cada tarea $j_{kf}(k)$, consumiendo un tiempo ininterrumpido $t \in N$.

Definición 9. El costo de comunicación entre el trabajo y sus correspondientes tareas.

La evaluación en paralelo de los objetivos contrapuestos de la planificación y la asignación de los trabajos en un sistema multiprocesadores

El método propuesto en (Velarde A., 2013), parte de los supuestos siguientes:

- Se tienen las distancias simétricas entre procesadores, que representan los saltos que un mensaje debe ejecutar para llevar a cabo la comunicación entre dos procesos.
- Se tiene un conocimiento previo del grado de comunicación (costos de comunicación) entre: el trabajo y sus tareas, y entre las tareas mismas.

La funcionalidad del método descrito en (Velarde A., 2013), se divide en cinco fases: a) la comunicación entre el planificador y el asignador de las tareas a los procesadores, b) la selección dinámica de trabajos de la cola de espera, utilizando la política de planificación Servicio de Orden Aleatorio ROS (por sus siglas en Ingles Random Order of Service) (W. Rogiest, 2015), c) la evaluación de la aptitud de las soluciones creadas, d) la generación de las nuevas poblaciones, y e) la asignación de los mejores individuos a la malla de procesadores.

La fase, de la evaluación de la aptitud de las soluciones creadas, se divide en las siguientes etapas:

1. Se selecciona un conjunto de trabajos de la cola de espera, que representa una solución factible de asignación de los trabajos a la malla de procesadores. Este conjunto de trabajos se construye de forma dinámica, mediante ROS, con la selecciona de diferentes trabajos de la cola de espera, mediante un ciclo condicionado hasta que una de tres condiciones se cumple: todas las tareas han sido seleccionadas al menos una vez, se alcanza el umbral de tareas seleccionadas o se ha agotado el número de procesadores libres en la malla.

ISSN-2444-5002

2. El individuo creado, es evaluado en los tres criterios siguientes:

    a. el porcentaje de la fragmentación externa,

    b. la carga de la comunicación entre el trabajo y las tareas y entre las tareas mismas, y

    c. el número de tareas que se extraen de la cola de espera y que son asignadas a la malla de procesadores

3. Los mejores individuos son extraídos de la población.

4. Repetir desde el paso 1, hasta que el número de poblaciones generadas y evaluadas alcanzan el umbral establecido.

El método descrito en (Apolinar, 2016), evalúa los criterios de utilization, throughput, mean turnaround time, waiting time and the total execution time, y se compara con la política de planificación más comúnmente usada, el primer trabajo que entra, es el primer trabajo que sale FIFO (por sus siglas en Ingles First Input First Output). Debido a la carga computacional que el sistema realiza, al evaluar cada uno de los individuos que constituyen las poblaciones, los resultados de los criterios muestran que el método propuesto apenas supera a la filosofía FIFO, y cuando las cargas superan los 10 000 trabajos encolados, las técnicas muestran resultados iguales.

En (Velarde, 2015), para superar estos resultados adversos, se realiza un análisis de los objetivos previo a las asignaciones de los trabajos a la malla de procesadores. Este análisis evalúa de forma secuencial los tres criterios para cada individuo: el porcentaje de la fragmentación externa, las cargas de comunicación y el número de tareas que se extraen de la cola de espera. El sistema espera a la emisión del resultado del primer criterio, para avanzar a la evaluación del segundo criterio, y espera la emisión del resultado del segundo criterio para evaluar el tercero. El incremento en los tiempos de espera para estas evaluaciones es muy alto, provocando la demora en la búsqueda del individuo que representa la mejor asignación de los trabajos a la malla de procesadores.

Considerando las demoras en los tiempos de ejecución del algoritmo, en este trabajo, se propone un procesamiento en forma paralela de los tres criterios que son evaluados antes de la asignación de las tareas a la malla de procesadores, y considera a su vez que cada procesador está constituido por un conjunto de núcleos de procesamiento, en donde pueden ser asignadas el mismo número de tareas para su procesamiento. Este trabajo, no hace uso de las características específicas de los niveles de jerarquía de memoria cache que constituyen el sistema hardware (arquitectura subyacente), sino únicamente la programación sobre este tipo de sistema.

Funcionalidad del sistema propuesto en este trabajo de investigación

La funcionalidad del sistema propuesto se divide en cinco pasos principales, de la forma siguiente:

1. Un núcleo de procesamiento, al que denominamos núcleo 0, es el responsable de ejecutar el método hasta la etapa 2 (descrita en la sección a anterior), cuando el individuo es creado.

2. En el paso 2, cuando se hace necesario evaluar el individuo creado en los tres criterios (objetivos a evaluar), cada evaluación es enviada a cada uno de los núcleos de procesamiento dentro de la maquina servidor.

3. Cada núcleo de procesamiento, es responsable de la evaluación de un objetivo: el porcentaje de fragmentación externa, la carga de comunicación y el número de tareas que son extraídas de la cola de espera.

4. Los resultados obtenidos de cada evaluación son recibidos en el núcleo 0, para su comparación y evaluación.

5. El núcleo 0 determina la mejor asignación en la malla de procesadores, e invoca al procedimiento para la asignación del trabajo con sus subtareas a la malla de procesadores.

Esta división de trabajos, permite al método evaluar de forma paralela los objetivos que define la mejor asignación de los trabajos a la malla de procesadores.

Experimentaciones

Las experimentaciones realizadas utilizan la política de planificación FIFO, en comparación con el método paralelo propuesto.

La política de planificación FIFO, es considerada una de las políticas más utilizadas en la planificación de las tareas.

Debido a la complejidad del algoritmo, y el tiempo necesario para realizar las experimentaciones con ambos método, se consideran únicamente dos métricas de desempeño del sistema: el tiempo total de ejecución y el tiempo de espera de las tareas en la cola de espera. La primera métrica, permite contabilizar en segundos, el tiempo que ambos métodos gastan desde que ingresa un trabajo con sus tareas a la cola de espera, hasta que terminan su ejecución dentro del sistema. Esta métrica nos permite evaluar, si el método propuesto ofrece menores tiempos de espera a los trabajos que son encolados o las tendencias de tiempo, son muy similares; en el caso de que los tiempos tiendan a ser similares, el método propuesto se considera inoperable e innecesario.

La métrica del tiempo de espera de los trabajos en la cola de espera, es el tiempo que un trabajo esperará para ser atendido por el asignador. Para el caso de nuestras experimentaciones, esta métrica nos permite observar el comportamiento del algoritmo ROS, con ambos métodos.

Los parámetros que se utilizan en las experimentaciones, son los siguientes:

1. El tamaño de la malla. Establece el tamaño de la malla y por consiguiente el número de procesadores en el sistema objetivo. Para este caso de estudio, consideramos únicamente mallas cuadradas.

ISSN-2444-5002

2. El número de trabajos que el sistema acepta para su ejecución. Es el número total de trabajos que el sistema procesará, llamado también la carga total del sistema.

3. Numero de tareas por cada trabajo encolado. Es el número de tareas que constituyen cada trabajo que es ingresado a la cola de espera.

4. Tiempo de ejecución para cada tarea. Es el parámetro que define el número de segundos que la tarea permanecerá dentro de la malla, y se constituye por la sumatoria de los segundos de cada una de las subtareas que constituyen la tarea.

5. Capacidad de la cola de espera. Es el número de tareas que acepta el waiting queue para su procesamiento, y el número de subtareas que cada tarea podrá contener.

6. Numero de tareas que el sistema buscará dentro de la cola de espera. Se define como el número de tareas que el algoritmo buscará en el waiting queue utilizando el método de planificación ROS. Este número de tareas lo determinan las condiciones de paro del algoritmo: si las tareas en la cola de espera han sido seleccionadas al menos una vez o si el número de procesadores libres en el tiempo t ya fue cubierto.

7. Numero de fenotipos o individuos por población que serán creados. Es el parámetro que define el número de individuos que conforman cada una de las poblaciones que construye el algoritmo para determinar el mejor individuo (conjunto de tareas que se asignaran a la malla de procesadores)

8. Numero de poblaciones que serán creadas. Se define como el número de poblaciones que el sistema genera para extraer los mejores individuos y estimar el modelo probabilístico.

9. Numero de núcleos en los que se procesaran las tareas. Se define como el número de núcleo que se utilizaran en la evaluación de los objetivos.

Las cargas del sistema tiene una variación de 2,500, 5000, y 10000 trabajos en ejecución, donde cada trabajo está constituido por un máximo de 100 tareas. Los resultados de las experimentaciones se muestran en las siguientes gráficas de desempeño del sistema.

La gráfica 1, muestra la métrica de desempeño del tiempo total de ejecución, que compara al método paralelo propuesto, con la política de planificación FCFS. Los resultados obtenidos con esta métrica, permiten observar que los tiempos de ejecución se disminuyen con el método paralelo, cuando las cargas de trabajo son superiores a 250 trabajos en la cola de espera. Con este volumen de carga de trabajo, se ha notado un desprendimiento en la igualdad de resultados, en los tiempos de ejecución como los que se muestran en (Apolinar, 2016).

ISSN-2444-5002

Los tiempos totales de ejecución, pueden considerarse como un factor de comparación, pero es necesario considerar otros factores que durante la ejecución de las tareas es necesario considerar, por ejemplo a) el número de tareas que los trabajos contienen, b) el orden de llegada de los trabajos, a la cola de espera, y c) si el sistema soporta trabajos con diferentes tipos de prioridad para su ejecución. Para los resultados obtenidos en las primeras experimentaciones, se han considerado un máximo de 100 tareas por trabajo, y considerando que cada tarea puede ingresar a un procesador y permanecer en éste hasta su finalización; para el caso del orden de llegada de los trabajos, éstos pueden ser aceptados toda vez que existen espacios vacíos (uno o más trabajos han sido ingresados a la malla de procesadores), sin restricciones; y no se aceptan ningún tipo de prioridades.

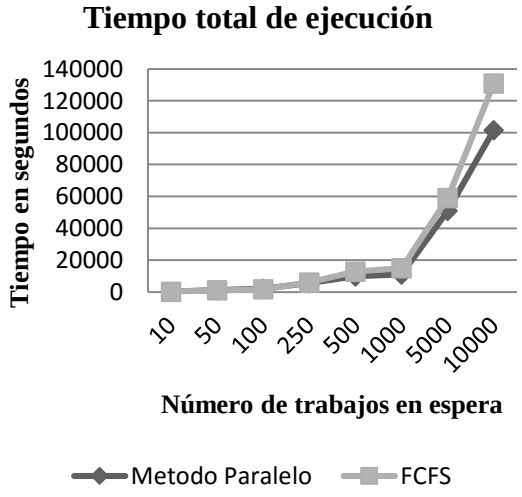

Gráfico 1 Gráfica comparativa de los métodos, en la métrica de desempeño: tiempo total de ejecución.

La grafica 2, muestra los tiempos que los trabajos esperan a ser programados para su ejecución en el sistema de procesadores.

Observaciones: con el método paralelo las tareas esperan menos tiempo en la cola de espera, debido a la aceleración en la velocidad de procesamiento de los objetivos evaluados; al ser enviadas las evaluaciones a diferentes núcleos de procesamiento, los resultados son acogidos en el núcleo de procesamiento 0, para su evaluación y selección. Al extraerse los trabajos con mayor prontitud de la cola de espera, estos permanecen menos tiempo esperando ser asignados. La técnica ROS extrae de manera aleatoria las tareas de la cola de espera, lo que provoca que todas las tareas tengan la misma probabilidad de ser seleccionadas y puedan ser asignadas a los procesadores, considerando que es posible asignar el trabajo en el número de procesadores que se encuentren libres, al momento de la selección. En forma contraria la política de asignación FIFO, permite únicamente tomar la tarea que se encuentra en la cabeza de la cola, produciendo que aunque existan procesadores libres, estos no pueden ser asignados mientras no exista el número de procesadores solicitados por la tarea; pero, la bondad más importante es que no tiene el problema de la inanición de tareas (para una referencia sobre la inanición de tareas el lector puede consultar (Velarde A., 2013)).

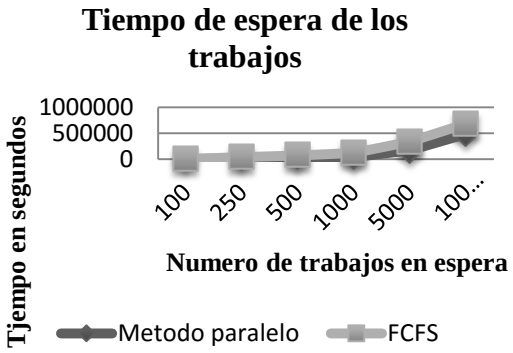

Figura 2 Gráfica comparativa de los métodos, en la métrica de tiempo de espera de las tareas.

Conclusiones

Los sistemas de cómputo paralelo son la herramienta fundamental para acelerar la velocidad de procesamiento de los datos. Las limitantes de procesamiento de grandes volúmenes de información han sido superadas con arquitecturas de cómputo paralelo. Pero como cada cualquier tecnología existente, presenta dificultades en su desarrollo, y una de estas, en los sistemas multiprocesadores es la planificación y la asignación de trabajos.

Este trabajo, aborda el problema de la planificación y la asignación de las tareas tomando como referencia los dos trabajos siguientes: primero, el análisis de los objetivos contrapuestos que se involucran en la planificación y la asignación de las tareas en un sistema de multicompuutadoras, y segundo: planificación y asignación de tareas en un sistema de multicomputadoras como un problema multiobjetivo y su resolución utilizando programación evolutiva. Utilizando los algoritmos propuestos, se determinó que, tienen segmentos de código que pueden ser paralelizados, utilizando las librerías OpenMP. Aunque los dos trabajos base mencionados, evalúan 5 criterios de desempeño de los sistemas multiprocesadores, únicamente se han evaluados dos de ellos en esta investigación: el tiempo total de ejecución y el tiempo de espera de las tareas, que servirán como base para determinar la utilidad de este proyecto de investigación.

Los resultados obtenidos en los experimentos realizados, demuestran parcialmente la factibilidad del proyecto: se mejoran ambos tiempos, lo que demuestra que al realizar una evaluación en paralelo los objetivos contrapuestos, se obtiene una ganancia en los tiempos de respuesta a los trabajos encolados, y por ende un procesamiento más ágil. Si bien es cierto que, faltan muchas experimentaciones por realizar, pero sirvan las presentes, como los resultados incipientes de este método para colaborar en la solución del problema propuesto.

Trabajos Futuros

En esta sección se describen los trabajos futuros que se pretenden desarrollar, en una segunda fase de este proyecto. Así, estas actividades versan en dos vertientes: a) considerar las subsecuentes métricas de desempeño de un sistema planificador, y que no son consideradas en este proyecto. Aunque las métricas evaluadas en este trabajo muestran resultados promisorios en la planificación y asignación de trabajos al sistema multiprocesador, es necesario evaluar otras métricas tales como: rendimiento del procesamiento, tiempo de permanencia promedio y coeficiente de respuesta; b) realizar la migración de las tareas para su ejecución, en cada uno de los procesadores asignados por el planificador, dentro de la agrupación única y homogénea.

Referencias

Apolinar, V. M. (2016). Planning and Allocation of Tasks in a Multiprocessor System as a Multi-objective Problem and its Resolution using Evolutionary Programming.

International Journal of Advanced Computer Science and Applications, 349-360.

Frias, F. C. (2006). An efficient syncronization model for OpenMP. *Journal of Parallel and Distribution Computing 66 Elsevier*, 1359-1365.

G. Feitelson, L. R.-9.-1. (2006). *Theory and Practice in Parallel Job Scheduling.* Recuperado el 2016, de http://www.cs.huji.ac.il/~feit/parsched/jsspp97/p-97-1.pdf

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Massachusets: Addisson –Wesley.

H. Hussain, e. a. (2013). A survey on resource allocation in high performance distributed computing systems. *Parallel Computing 39 Elsevier*, 709–736.

H. Jin, D. J. (2011). High performance computing using MPI and OpenMP on multi-core parallel systems. *Parallel Computing 37 Elsevier*, 562–575.

Martinez, A. V. (2015). Liebres Inteligentes: Sistema de Multicomputadoras para el procesamiento paralelo de aplicaciones científicas. *Revista de Tecnología e innovación ECORFAN Bolivia*, 454-463.

P. F. Dutot, T. N. (2009). Scheduling Parallel Task Graphs on (Almost) Homogeneous Multicluster Platforms. *IEEE Transactions on Parallel and Distributed System*, 940-952.

R. Chandra, L. D. (2001). *Parallel Programing in OpenMP*. Washington: Academic Press.

Sathe, M. P. (2011). Architecture Aware Programming on Multi-Core Systems. *International Journal of Advanced Computer Science and Applications*, 105-111.

Sinnen, O. (2007). *Tasks Scheduling for Parallel Systems.* Washington: Wiley Series.

Velarde A., P. d. (2013). Planning and Allocation Tasks in a Multicomputer System as a Multi-objective Problem. *Advances in Intelligent Systems and Computing 227. EVOLVE 2013. International Conference held at Leiden University, July 10-13, 2013.* . Leiden, The Netherlands: Springer.

Velarde, A. (2015). Objective Analysis in Task Planning and Allocation of Multicomputer Systems. *Research in Computing Science RCS*, 23-39.

W. Rogiest, K. L. (2015). When Random-Order-of-Service outperforms First-Come-First-Served. *Operations Research Letters*, 504-506.

ISSN-2444-5002