CONCIENCIA TECNOLÓGICA No. 55, Enero-Junio 2018: 5-13

Generación de Grafos Dirigidos Acíclicos mediante Métodos Aleatorios para la Planificación de Tareas *

Generation of Directed Acyclic Graphs by Random Methods for Scheduling Tasks

Investigación

Dr. Apolinar Velarde-Martínez[1]
[1] TecNM/Instituto Tecnológico el Llano Aguascalientes, Departamento de Sistemas y Computación, Carretera Aguascalientes - San Luís Potosí Km. 18, El Llano, Aguascalientes, México.
Tels. (449)916-12-51 , e-mail: avelardem@gmail.com

Resumen

Los Grafo Dirigidos Acíclicos (DAGs, por sus siglas en inglés Directed Acyclic Graph) se utilizan en diferentes áreas de la investigación científica. En el área de la computación, sirven para representar tareas (programas) constituidas por subtareas subyacentes (subprogramas) que pueden ser ejecutados en diferentes procesadores en un ambiente distribuido. Los DAGs, se generan mediante diferentes métodos que los producen de forma aleatoria, para crear cargas sintéticas que facilitan probar nuevos algoritmos de planificación de tareas en sistemas de cómputo paralelo y distribuido. La planificación de tareas, es un problema que se ha abordado con diferentes técnicas de programación para optimizar los recursos de los sistemas de cómputo. Aunque son técnicas y métodos muy elaborados, en su mayoría carecen de un análisis previo a la planificación de los DAG. Por lo anterior, en este trabajo se describen seis métodos para la generación aleatoria de DAGs; se seleccionan dos de estos métodos para experimentar la generación de aleatoria de DAGs, con el objetivo de realizar un análisis previo de cada uno de los DAGs y posteriormente construir cargas sintéticas que serán utilizadas como estructuras de entrada, en un nuevo algoritmo para la planificación y la asignación de tareas, en un Sistema de Cómputo Heterogéneo Distribuido (SCHD). A los DAGs producidos, que constituyen la carga sintética, se evalúa la anchura del grafo, debido a que es una de las características más incidentes en el proceso de la planificación y la asignación. Los resultados obtenidos muestran que, cada uno de los métodos produce DAGs con anchuras de grafo totalmente disímiles.

Palabras clave: Grafo Dirigido Acíclico, Sistema de Computo Heterogéneo Distribuido (SCHD), Algoritmo para la planificación y asignación tareas en un SCHD, Tareas Paralelas.

Abstract

Directed Acyclic Graphs (DAGs) are used in different areas of scientific research. In the area of computing, they serve to represent tasks (programs) constituted by underlying subtasks (subprograms) that can be executed in different processors in a distributed environment. The DAGs are generated by different methods that produce them in a random way, to create synthetic loads that make it easier to test new task planning algorithms in parallel and distributed computing systems. The planning of tasks, is a problem that has been addressed with different programming techniques to optimize the resources of computer systems. Although they are very elaborate techniques and methods, most of them lack an analysis prior to the planning of the DAG. Therefore, in this work, we describe six methods for the random generation of DAGs; Two of these methods are selected to experiment with the generation of random DAGs, with the objective of carrying out a preliminary analysis of each of the DAGs and subsequently construct synthetic loads that will be used as input structures, in a new algorithm for planning and assignment of tasks, in a Distributed Heterogeneous Computing System (DHCS). To the produced DAGs, which constitute the synthetic load, the width of the graph is evaluated, because it is one of the most incidents characteristics in the planning and allocation process. The results obtained show that each of the methods produces DAGs with totally dissimilar graph widths.

Keywords: Directed Acyclic Graph (DAG), Heterogeneous Distributed Computer System (HDCS), Algorithm for scheduling and assigning tasks in a HDCS, Parallel Tasks.

Introducción

Los grafos son estructuras que representan las relaciones, las interdependencias y las características entre los objetos, y son utilizados en diferentes áreas

* Este proyecto está financiado por el Tecnológico Nacional de México TecNM. Identificador: 2018-2 110

de la investigación científica [1,2,3,4]. En el área de la computación paralela y distribuida, se utilizan para desagregar las distintas tareas que conforman la solución de un problema, y la forma en que se distribuyen estas tareas entre los procesadores mediante los algoritmos de planificación y asignación de tareas [5]; los grafos también permiten mostrar las dependencias, las restricciones de precedencia, los enlaces de comunicación, los costos de cálculo y los costos de comunicación de las tareas que constituyen la aplicación a ser ejecutada dentro del sistema de cómputo [2,5,6].

Para realizar pruebas de los nuevos algoritmos de planificación en los sistemas de cómputo paralelos y distribuidos, se producen los grafos de dos formas: mediante la generación aleatoria con algunos de los métodos ya existentes tales como [1], [3], [9], [10], [11], o algún otro que representa cargas sintéticas, o mediante la generación de los grafos de las aplicaciones reales [4]. La generación de cargas sintéticas, se produce de acuerdo a las métricas de desempeño del algoritmo de planificación que se desea analizar. Hasta donde los autores tienen conocimiento, no existe un algoritmo estándar que produzca los grafos con las características específicas que concuerden tanto con el algoritmo que se desea evaluar y, con el sistema subyacente, por lo que se hace necesario evaluar diferentes algoritmos de generación de grafos, con los nuevos algoritmos de planificación.

De esta forma, cuando se experimenta con un nuevo algoritmo de planificación y asignación de tareas, es necesario observar cuidadosamente cada una de las características de los DAGs (en este trabajo utilizamos las siglas DAG, por estandarización de uso en la literatura); estas observaciones, nos permiten evitar sesgos en los resultados que se obtienen cuando se realizan las pruebas de convergencia, velocidad, capacidad de asignación de los recursos y, las velocidades de transferencia de los nuevos algoritmos.

Por lo antes expuesto, los tres objetivos de este trabajo son: primero, describir seis métodos de generación de grafos y seleccionar dos de ellos, que todavía no han sido utilizados para la generación de cargas sintéticas para la planificación de tareas (según se validó, en la literatura estudiada); segundo, con los dos métodos seleccionados, producir conjuntos de DAGs y evaluar, analizar y obtener valores de la anchura de los grafos generados. Tercero, con los grafos generados, constituir las cargas sintéticas que validarán a un nuevo algoritmo para la planificación y la asignación de tareas en un sistema de cómputo heterogéneo distribuido; el nuevo algoritmo se expondrá en trabajos de investigación posteriores a éste.

Este trabajo está organizado de la siguiente forma: la sección de fundamentación teórica, presenta las definiciones básicas y los trabajos relacionados; las definiciones básicas son los términos básicos que se utilizarán en el desarrollo de este trabajo, y que permiten al lector, ubicarse de mejor manera en la temática tratada; los trabajos relacionados, son descripciones de las características principales de los trabajos relacionados con esta investigación. En la sección de materiales y métodos, se describen seis métodos para la generación de grafos, tratados en este trabajo y que justifican la utilización de los dos métodos seleccionados para generar los grafos dirigidos acíclicos aleatorios. En la sección justificación de la generación aleatoria de grafos, se describe porque es justificable generar cargas aleatorias para validar nuevos algoritmos de planificación de tareas; esta descripción está basada en la literatura existente. La sección, modelado de tareas paralelas con grafos, explica la forma en que se utilizan los grafos para modelar las tareas paralelas que se procesan en los SCHD. La sección aplicación de los DAGs en el problema de la planificación de tareas en los SCHD, expone la forma en que los DAGs se utilizan para planificar las tareas y asignar a éstas los recursos que requieren en los SCHD. La sección de resutados, describe en dos subsecciones los resultados obtenidos con los métodos aplicados para la generación de las cargas sintéticas. En una subsecuente sección, se muestran las conclusiones de este trabajo y finalmente, los trabajos futuros que son consecuencia de esta investigación.

Fundamentación teórica

Un grafo es una estructura que representa relaciones e interdependencias entre objetos, y las características que los relaciona.

Ejemplos de las aplicaciones de grafos pueden ser las relaciones de parentesco entre personas, la estructura del diseño de páginas web, estructuras de datos básicos de aplicaciones para visualización de información [1], así como herramienta de modelado en varios campos (ciencias sociales, informática y biología) [2], y en la representación de programas paralelos que se modelan en la planificación de tareas en los Sistemas de Cómputo de Alto Rendimiento (HPCS, por sus siglas en inglés, High Performance Computing System) [7].

Dada la variedad de grafos en la literatura existente [12], en esta sección definimos un tipo especial de grafo denominado grafo dirigido ponderado acíclico (el cual referimos en las secciones siguientes como DAG), utilizando para ello las definiciones siguientes, extraídas de la literatura existente [12,15].

Definición 1. Un grafo *G* es un par *G = (V,E)* consistente de un conjunto finito V≠∅ y un conjunto *E* de dos elementos subconjuntos de *V*. Los elementos de *V* son llamados vértices. Un elemento $e = \{a,b\}$ de *E* es llamada una arista con vértices finales *a* y *b*. Se dice que *a* y *b* son incidentes con *e* y que *a* y *b* son adyacentes o vecinos uno de otro, y se define como e = *ab* o *a*e*b*.

Definición 2. Para determinar la relación que existe entre la información de los vértices (que las conexiones no modelan) se define un dígrafo. Un dígrafo, existe cuando el conjunto de conexiones $A = A(G)$ es dirigido, es decir, se distinguen entre las conexiones $e_{ij} = (v_i, v_j)$ y $e_{ji} = (v_j, v_i)$, entonces el grafo $D = (V, A)$ se denomina grafo dirigido o dígrafo.

Definición 3. Ahora, si entre las conexiones el dígrafo tiene relacionado un número $T(v_i, v_j)$ que representa el costo de comunicación entre el vértice v_i y el vértice v_j, se tiene un grafo ponderado. Un grafo ponderado, es un par (G,W) donde G es un grafo y W es una función $W:E \rightarrow R^+$, de esta manera el peso de una conexión e es $W(e)$. El peso del grafo es $W(G) = \sum_{e \in E} W(e)$.

Definición 4. Un grafo que no tiene ciclos en conexiones paralelas es decir no tiene conexiones de la forma: $e_{v_i v_i}$ se denomina un grafo acíclico.

Definición 5. Dado un DAG $G = (V,E)$, donde cada nodo $v \in V$ tiene una anchura positiva W_v; una división por capas o niveles de G, (también llamada, estratificación de G) es una partición de su conjunto de nodos V dentro de subconjuntos disjuntos $V_1, V_2, ..., V_h$, tal que si $(u,v) \in E$ donde $u \in V_i$ y $v \in V_j$ entonces $i > j$. Un DAG con una estratificación o división por niveles, es llamado un dígrafo estratificado.

Definición 6. La altura de un dígrafo estratificado, es el número de niveles *h*, del dígrafo.

Definición 7. La anchura de un nivel V_k es tradicionalmente definido como, $w(Vk) = \sum_{v \in v_k} w_v$ y la anchura de un dígrafo estratificado (dividido en capas o niveles) es definido por la ecuación $w = max_{1 \leq k \leq h} w(V_k)$.

Trabajos relacionados

En la literatura no se han encontrado trabajos que sigan una metodología de investigación como la que se propone en este trabajo: producir DAGs con métodos de generación aleatorios, extraer las características de los DAGs, evaluar estas características para obtener los resultados de cada una de ellas y, realizar un análisis previo a la planificación de los DAGs en un sistema de cómputo heterogéneo distribuido.

Lo que se ha encontrado en la literatura son tres tipos de trabajos que conforman nuestra investigación: primero, trabajos que destacan la importancia de las características de los DAGs en la planificación de tareas [4,7], segundo, trabajos que utilizan los DAGs como estructuras paralelas para la planificación de tareas en los SCHD [5,6,7] y, tercero, trabajos de investigación que desarrollan algoritmos para la generación aleatoria de DAGs [1,3,8,9,10,11].

En base al enfoque que se realiza en la investigación aquí propuesta, se mencionan y resumen brevemente los trabajos que destacan la importancia de las características de los DAGs, y seis de los trabajos que, por su importancia práctica en la literatura, se han propuesto para desarrollar los algoritmos que generan los DAGs.

Algunos de los trabajos que destacan la importancia de observar las características y los parámetros de los DAGs son [4,7]. En [4], las características definidas y observadas son:

- la ruta más larga (The longest path) definido como la trayectoria con longitud máxima, es decir, la trayectoria con el número más grande de nodos en un grafo,
- la distribución de aristas dirigidas (Distribution of the Out-degree) que se define como número de aristas dirigidas en el grafo que comienzan en el vértice v,
- el número de aristas (Number of edges), que representa la cantidad de aristas de los grafos generados.

En [7], los parámetros que son observables de los DAGs son la anchura y la densidad, y se definen de la siguiente manera: la anchura del grafo determina el máximo paralelismo en el DAG, esto es, el número de tareas en el nivel más largo. La densidad del grafo, indica el numero de aristas entre dos niveles del DAG.

Los trabajos de investigación que desarrollan algoritmos para la generación aleatoria de DAGs son [1,3,8,9,10,11]. En las siguientes subsecciones se describen estos seis trabajos que se han propuesto en la literatura, para la generación aleatoria de grafos. La descripción aquí realizada permite contrastar los parámetros utilizados en cada uno de ellos, entender la función de probabilidad utilizada y detectar las características de los DAGs generados.

Cabe mencionar que, métodos tales como: Erdös-Rĕnyi, nivel por nivel, Fan-in/Fan-out y Random Order ya han sido utilizado para el problema de la planificación de tareas en sistemas heterogéneos distribuidos, los resultados obtenidos con estos métodos pueden verse en [7]. Pero es de señalarse que, en este trabajo únicamente se generan los DAGs, y no se realiza un análisis previo de éstos, ni una extracción

de características antes de la planificación de las tareas.

El objetivo de investigar y analizar los métodos generadores de DAGS es debido a que, hasta donde los autores tienen conocimiento, no existe un procedimiento de generación de grafos aleatorios estándar, que representen programas paralelos a ser procesados por el SCHD, con una independencia absoluta de las métricas de desempeño que son utilizadas en los SCHD [4].

Por lo anterior expuesto, en este trabajo utilizamos dos métodos para la generación aleatoria de grafos, diferentes a los ya expuestos y utilizados en la literatura para la planificación de tareas, pero buscando también que los grafos generados, posean características semejantes a las que tienen los grafos de las aplicaciones paralelas del mundo real, como los expuestos en [4]; y la característica que se elige para observarse en los DAGs generados con los dos métodos, en este trabajo de investigación, es la anchura del grafo.

Materiales y métodos

1. Los métodos Erdös-Rĕny

Los métodos Erdös-Rĕnyi son dos modelos matemáticos simples, elegantes y generales [3], considerados los métodos más populares para la generación aleatoria de grafos [4]. El primer modelo, denotado como $\Gamma_{v,e}$ elige un grafo uniformemente al azar del conjunto de grafos con v vértices y e aristas [8]. La característica principal de este método es la generación de grafos aleatorios, con un número fijo de aristas [4]

El segundo método, denotado como $\Gamma_{v,p}$ elige un grafo uniformemente al azar del conjunto de grafos con v vértices, donde cada arista tiene la misma probabilidad p de existir [8]. De este método se pueden destacar las siguientes propiedades [4]:

- Cuando el valor de v es lo suficientemente grande, el número de aristas en los grafos generados tiende a $p\binom{n}{2}$.
- Existe una alta probabilidad de generar un subgrafo débilmente conectado con la mayoría de los vértices, si np tiende a una constante mayor que 1 y no existe ningún otro componente conectado con más de $0(\log n)$ nodos.
- Si $p > \frac{(1+\epsilon)\ \mathrm{In}\ n}{n}$ entonces es altamente probable que el grafo generado no tendrá vértices aislados.

2. El método nivel por nivel

Es un método diseñado específicamente para la validación de heurísticas de planificación [9]. Se basa en el concepto de los niveles; los niveles del grafo son las capas que forman la estratificación del grafo, formadas por las aristas que van de un vértice a otro [13,14]. Este concepto establece que, si existe una arista desde el nivel a al nivel b, entonces no existe una ruta de un vértice en b en a un vértice en a. las aristas se crean con probabilidad p exactamente como en el método de Erdös-Rĕnyi $\Gamma_{v,p}$.

La utilidad práctica de este método es debido a la posibilidad de limitar el tamaño de la ruta crítica del grafo, cuando se limita el valor de la variable k en el algoritmo.

3. El método Fan-in/Fan-out

Propuesto en [10], construye cada grafo incrementalmente, permitiendo un control sobre las propiedades de los vértices, o sobre la estructura general del grafo, mediante las variables out-degree e in-degree.

El método hace uso de las fases fan-in, fan-out como operaciones para expandir y contraer el grafo, lo que permite simular las fases de dispersión/contracción que se diseñan en las aplicaciones paralelas [4].

4. El método Random Order

Propuesto en [11], utiliza propiedades de la rama de las matemáticas denominada teoría del orden, para analizar y generar grafos aleatorios. Su funcionamiento se basa en la generación de conjuntos aleatorios ordenados parcialmente, que se usan para generar los grafos de tareas.

El concepto fundamental del método, se basa en crear un orden parcial por la intersección de varios ordenes totales (total orders).

5. El método de la cadena de Márkov

Este método, está basado en una cadena de Márkov para generar uniformemente dígrafos acíclicos aleatorios de un tamaño dado [1]. Este método, propuesto inicialmente para aplicaciones de visualización de información, busca producir dígrafos aleatorios acíclicos con:

- Un número prescrito de vértices uniformemente al azar iniciando desde el grafo vacío.
- Grado total acotado o grado de vértice acotado.
- Una forma de controlar la densidad de las aristas de los grafos resultantes.

Este algoritmo utiliza la siguiente propuesta de desarrollo: Sea $V = \{1...,n\}$ que denota el conjunto de vértices subyacentes del grafo considerado. Se define una cadena de Márkov M, con espacio de estado de todos los dígrafos acíclicos sobre el conjunto de vértices V. Una cadena de Márkov está completamente determinada por su función de transición, prescribiendo la probabilidad de que la cadena va de un estado dado a cualquier otro estado posible. Para este caso, la función de transición es como sigue:

Una posición consiste en un par ordenado (i,j) de distintos vértices de V. Si X_t denota el estado de la

cadena de Márkov en el tiempo t , entonces X_{t+1} se selecciona de acuerdo con las reglas (a) y (b) descritas debajo.

Suponer que una posición *(i,j)* se selecciona uniformemente al azar.

a. Si la posición *(i,j)* corresponde a un arco e en X_t, entonces $X_t = X_t \setminus e$. Esto es, la arista e es borrada del grafo asociado con X_t.

b. Si la posición *(i, j)* no corresponde a un arco e en X_t, entonces X_{t+1} se obtiene de X_t al adicionar este arco, siempre que el grafo subyacente permanezca acíclico; de otra manera $X_{t+1} = X_t$.

El algoritmo obtiene las características principales de la cadena de Márkov: aperiódica e irreducible con una matriz de transición simétrica, conteniendo una distribución estacionaria limitante, uniforme en el conjunto de todos los dígrafos acíclicos sobre el conjunto de vértices de V.

Una propuesta de mejora al algoritmo de la cadena de Márkov ha sido propuesta en [2]. En este trabajo de investigación, el algoritmo es ligeramente modificado y usado para generar dígrafos acíclicos simplemente conectados, uniformemente al azar. Este algoritmo se constituye por dos reglas T1 y T2, que aparecen en los siguientes párrafos.

Propuesta de desarrollo: $N \geq 2$ es un entero fijo, y $V = \{1, ..., N\}$ denota un conjunto finito de vértices. Considera el conjunto A de todos los grafos dirigidos acíclicos sobre V , esto es, los grafos que no contienen circuitos. En seguida, definimos la cadena de Márkov sobre el conjunto A. Debido a que el conjunto V de vértices es fijo, no se distingue entre un dígrafo en A y el conjunto de sus arcos. La transición en dos estados cualquiera en M, es dado de la siguiente forma:

X_t es el estado de la cadena de Márkov en el tiempo t. Suponga un par de enteros (i,j) que han sido dibujados uniformemente al azar desde el conjunto $V \times V$.

- Regla (T1). Si (i,j) es un arco en X_t este es borrado desde X_t. Esto es $X_{t+1} = X_t \setminus (i, j)$.
- Regla (T2). Si (i,j) no es un arco en X_t entonces:
    - Este es adicionado a X_t siempre que el grafo resultante sea acíclico. Esto es, $X_{t+1} = Xt \cup (i,j)$.
    - En otro caso nada es hecho, esto es, $X_{t+1} = Xt$.

Al iniciar el algoritmo desde un grafo con un conjunto vacío de arcos, es posible aplicar iterativamente las reglas (T1) y (T2) para construir un dígrafo acíclico con una distribución casi uniforme.

Las características del algoritmo que demuestran los autores son:

- La probabilidad de que una transición vaya de un estado X a un estado Y ≠ X es O ó $\frac{1}{N^2}$.
- La generación de la matriz de transición como simétrica.
- La convergencia de la cadena de Márkov hacia la distribución uniforme.
- La irreductibilidad del estado de espacio M.

6. Un enfoque paralelo para la generación aleatoria de grafos sobre GPU's

Este método, propuesto en [3], busca resolver el problema del crecimiento exponencial del número de aristas en el proceso de generación clásico de los grafos, con el método Erdös-Rĕnyi. El esquema general de esta investigación se basa en una colección de tres algoritmos secuenciales, como sigue: el primer algoritmo denominado ER, es la implementación del proceso aleatorio del modelo de Gilbert [8]; el segundo algoritmo ZER, explota la disponibilidad de una fórmula analítica para el numero esperado de aristas en los grafos generados, que se puedan omitir en un enfoque geométrico; un tercer algoritmo, PreLogZER, se implementa para evitar el cálculo de logaritmos requeridos por el método propuesto. Los tres algoritmos secuenciales se escalan a un formato paralelo, que se programa en un ambiente de GPU.

Una vez descritos los seis métodos para la generación aleatoria de grafos, cabe destacar que, en este trabajo de investigación, se analizan y se prueba la ejecución de los métodos 5 y 6, la cadena de Márkov y el método de un enfoque paralelo para la generación aleatoria de grafos sobre GPU's; la ejecución de estos dos métodos nos permite observar, analizar y obtener datos, de una de las características principales de cada uno de los DAGs generados: la profundidad del grafo.

La justificación de seleccionar estos dos métodos, se debe a que en la literatura aún no han sido utilizados para la generación de cargas sintéticas, para la planificación y la asignación de tareas en los SCHD.

Los resultados con la utilización de los otros cuatro métodos en la planificación de tareas, sin la extracción de características, y que aquí se describen (métodos 1, 2, 3 y 4), pueden verse en [4].

Justificación de la generación aleatoria de grafos

La generación de cargas de trabajo aleatorias, para validar un nuevo algoritmo de planificación es justificable porque:

- Puede ayudar a encontrar un contraejemplo para el algoritmo [4]. Aunque el algoritmo sea correcto

teóricamente, los datos de entrada aleatorios pueden ayudar a encontrar errores en la implementación, o ayudar a identificar cuellos de botella en el desempeño.

- Ayuda a evaluar el desempeño del algoritmo en contextos no analizados teóricamente [4]. Permite predecir cómo se ejecutará el algoritmo en condiciones reales.
- Permiten predecir algunas de las propiedades que tienen los digrafos acíclicos [1].
- Es posible obtener DAG's distribuidos uniformemente, con grado total acotado o grado de vértice acotado [1].
- Es probable obtener, en un gran conjunto de ejemplos, todos los casos posibles o interesantes que deban ser probados o estudiados [2].
- Permiten sintetizar conjuntos de datos, con el objetivo de evaluar la eficiencia y la eficacia de los algoritmos [3].

Modelado de tareas paralelas con grafos

A medida que los sistemas de computación distribuida heterogénea (por ejemplo, clusters, Grids, Clouds, etc.), se vuelven comunes para satisfacer las demandas computacionales masivas, de ejecutar aplicaciones científicas complejas que comprenden múltiples tareas, el proceso de asignar estas tareas a múltiples recursos computacionales como procesadores y dispositivos de entrada y salida, conocido como planificación y asignación de recursos en SCHD, es importante para el rendimiento de cualquier aplicación a ser ejecutada en este tipo de sistemas.

En los últimos años, los DAGs han recibido mucha atención como resultado del creciente interés en modelar las aplicaciones científicas de flujo de trabajo [5].

Este modelado, en el SCHD permite mostrar:

- las dependencias entre las tareas [6],
- la transmisión de datos entre las tareas [5],
- las restricciones de precedencia entre tarea [4] [6],
- los enlaces de comunicación entre tareas, los costos de cálculo de cada una de las tareas [6] y
- los costos de comunicación entre las tareas [4, 6].

En términos generales, un DAG es un modelo genérico de un programa paralelo, que consiste de un conjunto de procesadores (nodos) entre los cuales, existen dependencias [4, 5, 6].

En la figura 1, réplica de la imagen propuesta en [6], se muestra un modelo genérico de una tarea. Considerando el esquema de esta figura como un DAG, las dependencias entre las tareas aparecen en un ordenamiento jerárquico, por ejemplo n_1 con un costo de cálculo de 2 se ejecuta antes de las tareas n_2, .., n_5. Las transmisiones de los datos entre las tareas son dados a través de los vértices, donde se indica el costo de comunicación entre tareas, por ejemplo, existe una transmisión de datos entre n_1 y n_2, cuyo costo de comunicación es de 4.

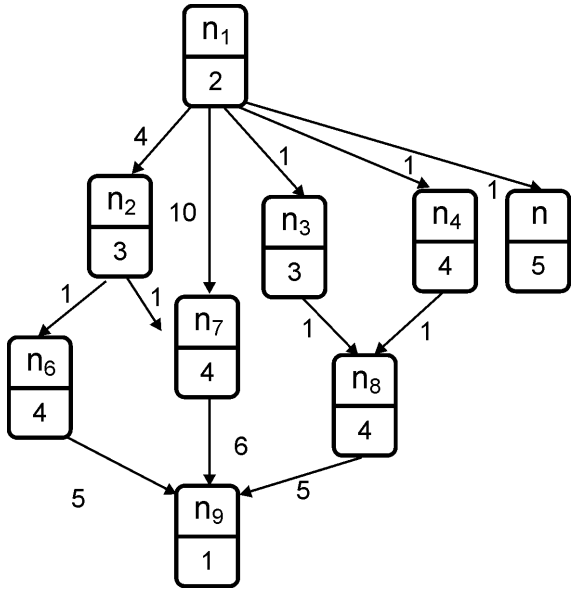


Figura 1. Grafo dirigido acíclico de una tarea [6].

Por la constitución de los DAGs, estos han sido utilizados para representar un esquema de un problema paralelo. En la siguiente sección, se define de una forma más general la manera en que los DAGs se aplican en el problema de la planificación y la asignación de tareas en un sistema de cómputo heterogéneo distribuido.

Aplicación de los DAGs en el problema de la planificación de tareas en los SCHD

Sin pérdida de generalidad y considerando las definiciones existentes en la literatura en esta sección, se define la forma en que los DAGs se aplican al problema de la planificación y la asignación de tareas en un SCHD.

Un DAG consiste en v nodos $n_1, n_2, ..., n_v$ que pueden ser ejecutados en cualquiera de los procesadores disponibles de un SCHD. Un nodo en el DAG representa una tarea, la cual, es un conjunto de instrucciones que deben ser ejecutadas secuencialmente sin derecho preferente en el mismo procesador. Un nodo tiene una o más entradas. Cuando todas las entradas están disponibles, el nodo se activa para su ejecución. Después de su ejecución, éste genera sus salidas. Un nodo sin padres es llamado un nodo de entrada, y un nodo sin hijos es llamado un nodo de salida. El peso en

un nodo es llamado el costo de cálculo (computation cost) de un nodo *ni* y es denotado por *w(ni)*.

El grafo también tiene *e* aristas dirigidos que representan un orden parcial entre las tareas. El orden parcial introduce una restricción de precedencia del DAG, e implica que si *ni→nj*, entonces, *nj* es un hijo el cual no puede iniciar hasta que su padre *ni* finalice y envía sus datos a *ni*. El peso, en una arista es llamado el costo de comunicación (communication cost) de la arista y es denotado por *c (ni, nj)*. En este costo se incurre si, *ni* y *nj* son planificados en diferentes procesadores y es considerado cero, si *ni* y *nj* se planifican en el mismo procesador. Para la estandarización, especificamos que un DAG tiene solo un único nodo de entrada y un solo nodo de salida.

Resultados y discusión

En este trabajo hemos utilizado dos métodos para la generación aleatoria de los DAGs: primero, el método 5, de la cadena de Márkov y el método 6, del enfoque paralelo para la generación aleatoria de grafos sobre GPU's.

De los DAGs generados con ambos métodos, se obtienen los datos de la anchura del grafo, para su análisis y observación de los efectos que tiene esta característica en la planificación de tareas en los SCHD.

El procedimiento para la realización de los experimentos, se describe en el siguiente algoritmo:

Entrada: el número de DAGs que se desean evaluar (No_DAG_evaluar)

Salida: el vector de niveles y método evaluado

1. *elección del método:*
    a. *cadena de Márkov*
    b. *Enfoque paralelo*
2. *Mientras (No_DAG_evaluar != contador)*
    a. *Generación del DAG*
    b. *Recorre el grafo generado*
    c. *Generación de los niveles del DAG mediante el método de la búsqueda en anchura*
    d. *Vector_niveles = N*úmero *de niveles del DAG*
    e. *Contador++;*
3. *Finmientras*
4. *Almacena el vector de niveles y método evaluado*
5. *Ordenamiento del Vector_niveles*
6. *El usuario compara los dos métodos evaluados según el parámetro: No_DAG_evaluar*
7. *El usuario describe los hallazgos*

Algoritmo 1. Algoritmo para la generación y evaluación de los DAGs.

Los algoritmos han sido codificados en lenguaje C, con las librerías OpenMP y ejecutados en una plataforma Unix, que corre en un servidor con procesador Quad Core marca HP Proliant.

Las gráficas que se presentan en esta sección, son representativas de los resultados que se obtienen de la ejecución repetitiva de los algoritmos con diferentes cargas (diferente número de DAGs generados). Por cuestiones de espacio, únicamente se muestran los resultados obtenidos en la secuencia de valores 5, 10, 15, 20, 25 y 30 para la variable N, y que son representativos del conjunto de pruebas realizadas.

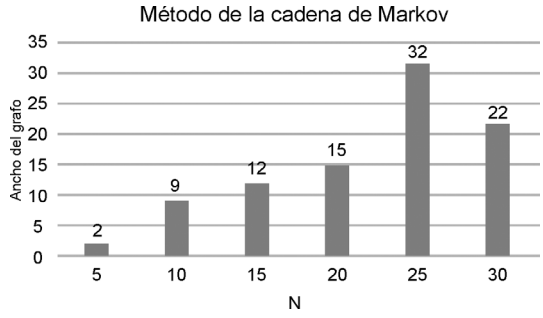
Textualmente, y para un mejor entendimiento de las gráficas presentadas, se explica que, las barras de las gráficas representan la anchura promedio del grafo generado con cada valor de N. Por ejemplo, para la gráfica 1, cuando la variable N que representa el conjunto finito de vértices, toma el valor de 5, el valor promedio de las capas generadas es 2. Cuando la variable N que representa el conjunto finito de vértices, toma el valor de 10, el valor promedio de las capas generadas es 9. Y así sucesivamente.

1. Experimentaciones con el algoritmo de Márkov

Para la ejecución del algoritmo de la cadena de Márkov, el parámetro N (el conjunto finito de vértices) toma los valores de 5, 10, 15, 20, 25 y 30. La probabilidad de que una transición vaya de un estado X a un estado $Y \neq X$ está dado por la fórmula $\frac{1}{N^2}$.

En las pruebas realizadas, se puede observar que la anchura generada en el DAG con este algoritmo, se produce en los niveles más altos, es decir, si el grafo se produce con 5 niveles, la anchura máxima se alcanza en el nivel 1 o 2, lo que implica que la asignación y la liberación de los recursos en el sistema heterogéneo distribuido, se hace en una etapa temprana de la ejecución del algoritmo de planificación. De esta forma, conforme se incrementa el valor de N, se incrementa el número de niveles, los cuales alcanzan un máximo de 32 cuando N=25, y disminuyen en las experimentaciones siguientes.

Los resultados obtenidos con este método, según las variaciones generadas en los valores de la variable N, se muestran en la gráfica 1.



Gráfica 1. Resultados obtenidos en la generación de DAGs, utilizando el método del enfoque paralelo para la generación aleatoria de grafos.

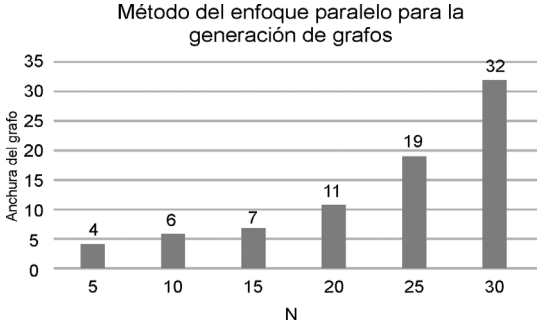
Dada la naturaleza aleatoria de la generación del DAG, es muy complicado determinar el comportamiento del algoritmo bajo diferentes condiciones, pero nos permite localizar los tiempos de utilización de los recursos, indicando claramente que cuando se utiliza el método de la cadena de Márkov, el diseño de las tareas que se planificarán, generarán requerimientos tempranos de recursos.

2. Experimentaciones con el método del enfoque paralelo para la generación aleatoria de grafos

Los parámetros puestos al algoritmo del enfoque paralelo de generación de grafos son: *m* que representa los puntos de ruptura de los intervalos, *E* el número máximo de aristas y *p* la probabilidad de inclusión.

Para generar un comparativo entre ambos métodos, se hizo variar el parámetro *N* con los mismos valores que el método de la cadena de Márkov, desde 5 hasta 30 con incrementos de 5 unidades. Los parámetros *m* y *p*, permanecieron constantes durante las pruebas realizadas. Para el parámetro *E*, se consideraron un número máximo de aristas N.

Los resultados obtenidos según las variaciones generadas se muestran en la gráfica 2.



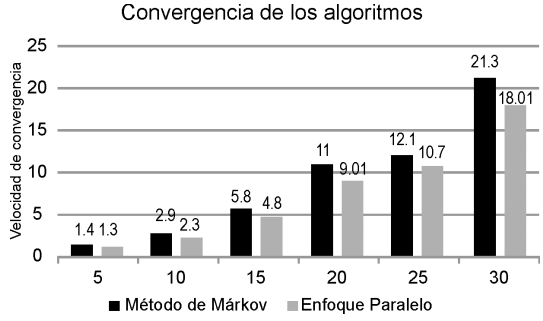
Gráfica 2. Resultados obtenidos en la generación de DAGs, utilizando el método del enfoque paralelo para la generación de grafos.

Los resultados obtenidos con este método muestran un adelgazamiento en la anchura del DAG, lo que se puede interpretar como la generación de aplicaciones paralelas con menos carga de recursos para el algoritmo que realiza la planificación.

Un punto muy importante en este algoritmo es el número de aristas producidas en el DAG. Mientras que el método de la cadena de Márkov genera pocas aristas, en este método es observable que en cada nivel se producen un numero sustancial de estas, lo que indica que las aplicaciones paralelas que se representan contienen altos índices de comunicación entre ellas, lo que permite evaluar los medios de comunicación del sistema distribuido.

Otra de las características que se han observado en los experimentos ha sido el tiempo de convergencia de los algoritmos propuestos, es decir el tiempo que necesita el algoritmo para generar la totalidad de los grafos.

El método de la cadena de Márkov, por su condición de ser un algoritmo secuencial, su tiempo total para la finalización de la generación de total de los grafos es ligeramente mayor, que el tiempo que necesita el método del enfoque paralelo para la generación de grafos, que nació con una condición de ser un algoritmo paralelo. En forma resumida en la gráfica 3 se muestran los tiempos de convergencia consumidos, por cada uno de los métodos.



Grafica 3. Tiempos de convergencia del método de la cadena de Márkov y el método del enfoque paralelo para la generación aleatoria de grafos.

El método de la cadena de Márkov, es utilizado para representar aplicaciones paralelas que requieren un gran número de recursos del sistema computacional, que serán utilizados en etapas tempranas del algoritmo; en tanto que, el segundo método facilita la generación de DAGs con altos requerimientos de comunicación en el sistema.

Finalmente, se destaca la importancia de ambos métodos para la generación de DAGs, que serán utilizados en los siguientes trabajos de investigación.

Conclusiones

En este trabajo, se propone y desarrolla una nueva metodología para abordar el problema de la

planificación y la asignación de tareas en un SCHD, considerando que las tareas que se planifican pueden ser representadas por grafos, y estos grafos a su vez pueden ser analizados para extraer características tales como la anchura, la ruta crítica, la densidad de borde, entre otras. La metodología aquí expuesta, propone en una primera instancia, la detección de la anchura de los DAGs producidos, utilizando dos métodos de generación aleatoria de grafos: el método de la cadena de Márkov y el método del enfoque paralelo para la generación aleatoria de grafos sobre GPU's.

El objetivo de detectar la anchura máxima, y el nivel en que ésta ocurre, es para determinar el momento en que el DAG producirá el paralelismo máximo, durante la planificación y la asignación de la tarea en el SCHD.

Las experimentaciones con estos dos métodos, han permitido analizar, visualizar y obtener datos que serán utilizados con un nuevo algoritmo de planificación. Los resultados obtenidos, muestran que los métodos generan grafos con características muy disimiles.

Trabajos futuros

Las investigaciones futuras y posteriores a este trabajo, están dirigidas a experimentar otras propiedades de los DAGs, tales como: la regularidad, la densidad y el parámetro de salto de la arista, con los mismos métodos expuestos en este trabajo.

Actualmente se desarrolla el algoritmo de planificación de tareas en un sistema de cómputo heterogéneo distribuido, el cual se diseña con la estrategia de metaheurística, y las pruebas de este algoritmo, se realizan con las cargas sintéticas generadas con los métodos descritos en este documento.

Referencias

[1] G. Melançon, I. Dutour, M. Bousquet-Mélou. (2001), Random Generation of Directed Acyclic Graphs. Electronic Notes in *Discrete Mathematics*. Vol 10, 202-207. Elsevier

[2] G. Melançon, F. Philippe. Generating connected acyclic digraphs uniformly at random. Disponible en http://arxlv.org

[3] S. Bressan, A. Cuzzocrea, P. Karras, X. Lu, and S. H. Nobari. (2013), An Effective and efficient parallel approach for random graph generation over GPUs. *Journal of Parallel and Distributed Computing*. 73, 303-316 Elsevier

[4] D. Cordeiro, G.Mounié, S. Perarnau, D. Trystram, J. M. Vincent, F. Wagner. Random graph generation for scheduling simulations. En http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8667

[5] W. Zheng, R. Sakellariou. (2013), Stochastic DAG scheduling using a Monte Carlo approach. *Journal of Parallel Distributed Computing,* 73, 1673–1689

[6] Y. K. Kwok, I. Ahmad. Benchmarking and Comparison of the Task Graph Scheduling Algorithms. En https://pdfs.semanticscholar.org/8dc6/1593e27e05f7c87c282e82084ce5ddd14592.pdf

[7] P. F. Dutot, T. N'Takpé, F. Suter, H. Casanova. Scheduling Parallel Task Graphs on (almost) Homogeneous Multicluster Platforms. *IEEE Transactions on Parallel and Distributed System,* Vol. 20, Issue 7.

[8] P. Erdös and A. Rěnyi. (1956), On random graphs I. *Publicationes Mathematicae* Debrecen 6:290-297.

[9] T. Tobita and H. Kasahara. (2002), A standard task graph set for fair evaluation of multiprocessor scheduling algorithms. *Journal of Scheduling*, 5(5):379-394.

[10] R. P. Dick, D. L. Rodes and W. Wolf. (1998), TGFF: Task Graphs for Free. In *Proceedings of the 6th International Workshop on Hardware/ Software Codesign*, pages 97-101, Washington, DC, USA, IEEE Computer Society

[11] P. Winkler. Random orders. Order 1(4):317-331, Dec. 1985

[12] D. Jungnickel. (2008), Graphs, Networks and Algorithms. *Algorithms and Computation in Mathematics*. Volume 5. Third Edition. Springer

[13] U. Rüegg, T. Ehlers, M. Spönemann, R. von Hanxleden. (2016), A Generalization of the Directed Graph Layering Problem. In: Hu Y., Nöllenburg M. (eds) *Graph Drawing and Network Visualization.* GD. Lecture Notes in Computer Science, vol 9801. Springer, Cham.

[14] P. Healy, N. S. Nikolov. (2002), A Branch-and-Cut Approach to the Directed Acyclic Graph Layering Problem. In: Goodrich M.T., Kobourov S.G. (eds) *Graph Drawing*. GD. Lecture Notes in Computer Science, vol 2528 pp. 98-109. Springer, Berlin, Heidelberg.

[15] P. Healy, N. S. Nikolov. (2002), How to Layer a Directed Acyclic Graph. In: P. Mutzel, M. Jünger, and S. Leipert (Eds.): GD 2001, LNCS 2265, pp. 16-30, 2002. Springer-Verlag Berlín Heidelberg

Recibido: 14 de octubre de 2017
Aceptado: 7 de diciembre de 2017