Algoritmo de planificación de orden parcial para optimizar la gestión de tareas en infraestructuras GRID

Holman Diego Bolívar Barón [a] y Mario Martínez Rojas [b]

R: 23012011 – A: 27042011

Resumen

El término *Grid Computing* creado por la comunidad de redes de energía eléctrica y adoptado por la de computación en malla nombra una infraestructura de hardware y software que proporciona fiabilidad, acceso constante y económico a una alta capacidad de computo que cada vez es más relevante en la investigación al ser una forma de virtualizar los recursos informáticos, a través de internet en aplicaciones distribuidas a gran escala. Debido a los diferentes tipos de recursos, es importante su gestión para la organización e interoperabilidad, razón por la cual se han diseñado modelos y algoritmos para la selección efectiva de recursos que permitan la ejecución de las tareas impuestas por los usuarios de la Grid. Debido a que la planificación busca control sobre la explosión combinatoria en la asignaciones de tareas sobre recursos disponibles, la presente investigación desarrollo un enfoque por medio de la descomposición de tareas en literales lógicos definiendo la sintaxis para la representación del problema, por medio de la especificación de estados utilizando el lenguaje de descripción de acciones, en primera instancia se abordó el problema global y luego se desarrollaron sub problemas con sub objetivos aplicando el principio de reducción con las interacciones negativas permitiendo el uso de enlaces causales eliminando la sobre estimación de búsquedas en múltiples objetivos, se logro optimizar el tiempo de ejecución en un 20% evidenciado en 10 escenarios diferentes de simulación.

Palabras clave:

Algoritmo
planificación
red informática

[a , b] Grupo de Investigación GISIC - Facultad de Ingeniería de la Universidad Católica de Colombia. Bogotá, Avenida Caracas No. 46-72 piso 4°. Contacto: hdbolivar@ucatolica.edu.co

Scheduling´s algorithm of partial order to optimize the management of task in Grid computing

Abstract

Key words

algorithms
computer networks
planning

Grid Computing was born of power grids and known as a hardware and software infrastructure that provides reliable, consistent and economical access to high computing power, is increasingly important in research to be a form of virtualize computing resources by Internet in large-scale distributed applications. Because different types of resources, it is important for the organization management and interoperability, which is why they have designed models and algorithms for effective selection of resources that allow the execution of the tasks required by users of the Grid. Because planning seeks control over the combinatorial explosion in assignments on available resources, this research developed an approach using task decomposition in literal syntax for defining logical representation of the problem, through the specification of states using the action description language, first addressed the overall problem and then developed problems with sub goals by applying the principle of reducing negative interactions with allowing the use of causal links by eliminating the overestimation of searches on multiple targets , was achieved optimize runtime evidenced by 20% in 10 different simulation scenarios.

Introducción

La investigación que a continuación se presenta hace parte del proyecto de identificación de modelos de complejidad para administrar infraestructuras *grid*, desarrollado con el auspicio de la Universidad Católica de Colombia y la Universidad Tecnológica de Panamá con apoyo de la Secretaría Nacional de Ciencia, Tecnología e Innovación (SENACYT) de la República de Panamá.

Aunque la noción de computación distribuida ha existido hace ya varias décadas, el concepto de *Grid Computing*, tuvo sus inicios a mediados de la década de los noventa del siglo XX para integrar las comunicaciones de las diferentes infraestructuras computacionales en las comunidades científicas más importantes del mundo. Los orígenes de la idea de *grid* para apoyar la investigación científica se remonta al pionero de Internet JCR Licklider (Waldrop, 2001), quien escribió un artículo, en el que sostenía que las computadoras deben permitir la colaboración de personas y equipos en la toma de decisiones y en el control de situaciones complejas, sin tener que depender de programas predeterminados.

El término *Grid Computing* fue adoptado de las redes de energía eléctrica y se entiende como una infraestructura de hardware y software que proporciona fiabilidad, acceso constante y económico a una alta capacidad de computo; es cada vez más relevante en la investigación al ser una forma de virtualizar los recursos informáticos, a través de internet en aplicaciones distribuidas a gran escala (Chunlin y otros, 2003). Ian Foster, concibe la *grid* como un intercambio seguro y coordinado de recursos que facilita el acceso a fuentes de datos geográficamente distribuidos, sirviendo de base para la resolución de problemas en la dinámica multi institucional dentro de organizaciones virtuales y a la vez es capaz de realizar trabajos de alto procesamiento con estos recursos (Xinjun y otros, 2006 y Nakada y otros, 2006).

Existe una amplia variedad de recursos computacionales tales como clústeres y supercomputadoras, incluyendo sistemas de pequeña escala como computadoras personales y estaciones de trabajo, también hacen parte, los dispositivos de visualización, instrumentos científico especiales, sistemas de almacenamiento y bases de datos. En resumen, todo lo que se encuentre disponible en la *grid* para ser utilizado ante un requerimiento de un subproceso o tarea es visto como un recurso *grid* (Hurtado y Rodas, 2009).

Los diferentes tipos de recursos dentro de la *grid* para su interoperabilidad y organización requieren gestión, razón por la cual en la última década, los investigadores e ingenieros se han dado a

la tarea de diseñar modelos y algoritmos para la selección efectiva de recursos que permitan la ejecución de las tareas impuestas por los usuarios, lo cual se conoce como el problema del planificador de tareas en la *grid* (Bolívar y Duran, 2010). Diferentes modelos propuestos para el planificador de tareas contemplan comunicación punto a punto, permitiendo el intercambio directo de información y flexibilidad en la planificación dentro de sistemas abiertos; equilibrando las cargas del sistema, mejorando la interoperabilidad entre compontes, la gestión de recursos, el rendimiento y fiabilidad; logrando un alto grado de escalabilidad (Tang y Zhang, 2011). Esta planificación se consigue a través del despliegue de tareas de un trabajo sobre los nodos del sistema, atendiendo a la necesidad de recursos, a la dependencia entre tareas, al rendimiento de factores como la concurrencia, al grado de paralelismo, al coste de la comunicación y al acceso a los recursos compartidos como la memoria (Leal y otros, 2009).

Los algoritmos estáticos se suelen utilizar para una planificación eficiente de trabajos que se van a ejecutar en algún momento en el futuro y para validar la eficiencia en el tiempo; también pueden ser útiles para el análisis de la heterogeneidad y para evaluar el desempeño de un sistema de programación dinámico después de su ejecución. Sin embargo este tipo de algoritmos de planificación en grid son difíciles de implementar porque los recursos son de propiedad de las diferentes organizaciones que tienen sus propias políticas y mecanismos de validación. La complejidad de las aplicaciones *grid* aumenta cuando los usuarios especifican restricciones como el tiempo y características especiales de los recursos a utilizar (Zhongzhi y otros, 2011). Ranganathan plantea que la efectiva elección de un algoritmo de programación tiene un impacto significativo en la planificación de tareas y la programación de datos dentro de la grid (Sang-Min y Jai-Hoon, 2003), mejorando notablemente el rendimiento de los procesos que permiten realizar la búsqueda y ubicación de los recursos.

Los desafíos más importantes que se deben afrontar para desarrollar una apropiada gestión de recursos se basan en la definición de una búsqueda consistente de recursos, que provean el tiempo mínimo de trabajo en la ejecución de las tareas programadas (Hamar, 2009) para el desarrollo a cabalidad de un trabajo, teniendo en cuenta los tiempos de ejecución y espera en los procesos que allí se desarrollan. Cada componente tiene sus propias funciones, recursos y pueden estar en organizaciones físicas diferentes (Cox y otros, 2007). Para la mayoría de sistemas *grid*, el problema real y concreto es coordinar la programación de recursos en organizaciones multi-institucionales, donde un algoritmo de planificación eficaz y eficiente es fundamental (Dong y Akl, 2006), (Sapra y Sunaina, 2010). Sólo con la ayuda de una política factible de planificación es posible que las *grid* puedan acelerar el procesamiento de trabajo y proporcionar servicios no

triviales a los usuarios (Chung y Chang, 2009). El problema de planificación de tareas, es complejo e importante en el equilibrio de la carga de todo el sistema para optimizar el uso de los recursos disponibles (Wu et al., 2011). Esta búsqueda, requiere un conjunto de factores que permitan explorar la grid y los nodos que se encuentran en ella sin importar su ubicación, implicando que sea casi imposible conocer el estado actual de cada uno de los nodos, es por esto que para entender mejor el *modus operandi* dentro de la *grid*, es necesario tener en cuenta sus principales componentes, entre los elementos clave se incluyen, la gestión de un nodo físico, la programación o gestión de recursos y la planificación de tareas. Estos factores influyen en la interoperabilidad de la grid, dividiéndose en dos tipos, los factores dinámicos, los cuales cambian con el tiempo, y los factores específicos de aplicación, los cuales intervienen durante la ejecución de los procesos (Sang-Min y Jai-Hoon, 2003). Los factores dinámicos son difícilmente previsibles ya que se transforman y cambian con el tiempo, la literatura define tres principales factores dinámicos dentro de ambientes *grid*:

ANCHO DE BANDA DE RED. El tiempo de transferencia de datos se encuentra en proporción al ancho de banda de red, es por esto que es necesario medir y pronosticar con precisión el ancho de banda porque el tiempo de transferencia de datos es fuertemente dependiente de la carga de la red y el tamaño de los datos en el entorno Grid es muy grande.

NÚMERO DE NODOS DISPONIBLES. En un entorno *grid*, el número de nodos disponibles cambia con el tiempo, lo que lo hace uno de los factores más importantes para la programación, pues el número de nodos disponibles en el sitio se decidirá de acuerdo a la carga de trabajo y la cantidad de procesadores destinados a los usuarios.

ATRIBUTOS DEL SISTEMA. Es casi imposible predecir con precisión el tiempo de ejecución de un determinado trabajo, más si sólo se tiene en cuenta el número de nodos disponibles en cada sitio, por esto es indispensable analizar que cada sitio cuenta con diferentes arquitecturas de sistema, con diferentes atributos que cambian dinámicamente. Por ejemplo, la velocidad del procesador, el tamaño de la memoria y el rendimiento E/S, los cuales pueden cambiar con el tiempo y variar unos respecto a otros.

Por el contrario, los factores específicos de aplicación, se basan en características únicas de cada aplicación. Los siguientes son los factores específicos de aplicación más importantes que se reflejan dentro de la *grid* (Sang-Min y Jai-Hoon, 2003):

EL TAMAÑO DE LOS DATOS DE ENTRADA. En general, en el entorno *grid*, gran cantidad de datos son analizados y replicados a los sitios dispuestos en diferentes puntos de la geografía mundial. Esto significa que si los datos no se replican en determinado sitio para el procesamiento de un trabajo, los datos deben ser traídos desde otros sitios. El realizar esta tarea de búsqueda y selección empeora el desempeño de la

grid, debido a que la velocidad de transferencia de los datos está limitada por el ancho de banda de red.

TAMAÑO DE LOS DATOS DE SALIDA. Si el trabajo de computación se lleva a cabo en un sitio remoto, los datos de salida que se producen deben ser transmitidos en el sitio local para que los usuarios puedan tener acceso a ellos. Cada aplicación intensiva de datos produce diferentes cantidades de datos.

Una herramienta efectiva para sopesar y analizar con detenimiento los factores que inciden en la interoperabilidad de la grid es la identificación de recursos y consultas a sus componentes. Para esta tarea es necesario valerse de referencias que son destinadas a cada uno de los recursos disponibles y se definen por la localización y el nombre del recurso. Las consultas que se realizan a los recursos se realizan con un método de comparación entre nombres referenciados, generando un mecanismo de identificación robusto que permite identificar con eficiencia la localización del recurso. A continuación se definen los principales métodos por los cuales se pueden realizar las consultas de los recursos dentro de la Grid (Chuburu y otros, 2008):

IDENTIFICADORES ÚNICOS. Si un recurso tiene un único nombre es más fácil de localizar, además se puede utilizar un identificador que permita definir y obtener un nodo donde se encuentre almacenada una referencia a la localización del recurso. Este tipo de identificadores se obtienen por medio de una función de *hash* sobre el texto plano del nombre del recurso. El principal problema que tiene este mecanismo es que no permite que las propiedades que cambian se vean reflejadas en el sistema de nombres y esto excluye a los recursos dinámicos, o por lo menos las propiedades dinámicas de los recursos no son reflejadas en el sistema de descubrimiento.

IDENTIFICACIÓN POR TEXTO PLANO. Como el nombre lo sugiere, se identifican los recursos por sus nombres. Esta alternativa, permite consultas más complejas sobre el nombre, como la utilización de expresiones booleanas, o consultas del tipo *suena como*, cosa que no se puede realizar sobre los identificadores únicos ya que estos no guardan relación con los mismos.

DIRECTORIOS. Son los sistemas de directorios construidos sobre espacios jerárquicos y escalables de nombres, como por ejemplo DNS y LDAP (Lightweight Directory Access Protocol). Muchos sistemas de descubrimiento utilizan algún esquema de directorio de nombres basado en la sintaxis de DNS o LDAP, o alguna variante de alguno de estos dos.

ATRIBUTOS. El sistema de nombres más poderoso es el de atributos, los recursos son descritos por medio de un número de atributos predeterminados que toman un valor y pueden ser utilizados para todo el rango de tipo de recursos que se mencionaba anteriormente, incluso para describir atributos de recursos dinámicos.

Los usuarios requieren de una optima formula que administre de manera funcional los recursos, adaptándose a las necesidades y aplicaciones específicas en las que son utilizados, por esto el usuario del recurso se define como la entidad que se encuentra conectada a la red y que hace uso de los recursos compartidos. Aunque hay que tener en cuenta que esta entidad podría ser un usuario humano, así como también una aplicación que requiere de determinados recursos para llevarse a cabo. El sistema de la *grid* para cumplir con los requisitos debe ser capaz de redirigir los procesos que se estén ejecutando en un componente en forma lenta, a otro que pueda cumplir con mayor eficiencia esta tarea permitiendo el desarrollo de la *grid* de forma optima y que el usuario este cómodo con el sistema en el que está inmerso.

De igual manera, la interoperabilidad de la *grid* juega un papel importante dentro del sistema, pues permite traducir y crear los protocolos que ayudan a establecer las normas necesarias para que las diversas organizaciones de la *grid* coexistan, de tal forma que los componentes heterogéneos no se vean excluidos y que el usuario y los procesos no noten cambio alguno (Hamar, 2009).

De otra parte, un planificador es un algoritmo de programación que se encarga de descubrir y gestionar los recursos de la *grid* permitiendo su utilización dentro de los diferentes componentes. Este algoritmo tiene una política de transferencia de trabajo que determina si hay o no necesidad de emigrar puestos de trabajo desde un nodo a otro nodo (Shan y otros, 2003). Para lograr esta tarea el planificador se vale de un sistema de vigilancia y migración de cargas de trabajo, el que permite administrar los recursos locales en el centro de recursos distribuido. Su principal objetivo es abordar los problemas de la alta tasa de fallos y disponibilidad de recursos dinámicos, con el fin de facilitar la adopción de la tecnología grid.

Comúnmente, un planificador global se encuentra disponible dentro de un intervalo de tiempo para todos los recursos necesarios y realiza una reserva previa en ese intervalo de tiempo, asegurándose de que todos los recursos estarán disponibles por un momento específico (Nakada y otros, 2006). Sin embargo, diversos recursos tales como computadoras, redes y puestos de almacenamiento dentro de la grid, son generalmente utilizados por los usuarios locales. Por esto, dentro de la grid la co-asignación se realiza por programadores de recursos, quienes tienen la tarea de proporcionar sus recursos, tanto para los usuarios locales, como para todos los usuarios a nivel mundial, por lo que esta tarea se debe cumplir de manera eficiente (Takefusa et al., 2007). Para resolver esta dificultad, los planificadores locales, básicamente asignan a cada puesto de trabajo un número determinado de recursos, pero sin perder el control de cada uno de los recursos locales, para no estropear esta tarea y fallar en el intento, es necesario implantar un método de reserva previa a los programadores

locales y globales. El sistema del planificador global asigna recursos por medio de procesos de co-asignación de recursos, de la siguiente manera:

RESERVA MANUAL DE RECURSOS Y EJECUCIÓN DE TRABAJOS. El usuario tiene la capacidad de reservar los recursos distribuidos a través de negociaciones por medio de métodos tales como, el correo electrónico y las llamadas recurrentes por medios telefónicos; logrando así conciliar una reserva con cada administrador de recursos para la disposición y utilización de los mismos. Esta estrategia es difícil de utilizar y para nada es funcional.

RESERVA MANUAL DE RECURSOS ADMINISTRADOS POR LOS PLANIFICADORES DE LOS RECURSOS. En este caso el usuario reserva los recursos de forma manual como en el anterior proceso, aunque aquí los administradores de recursos se valen de la ayuda de un planificador que configura una cola de solicitudes dependiendo de los requisitos solicitados por el usuario. Hecho esto solo basta que le envíen los trabajos requeridos para procesar a las colas de espera y listo. El proyecto Tera Grid en los EE.UU. adopta esta estrategia, que permite la gestión de los recursos en función de cada organización, pero se han reportado muchos errores en la configuración del planificador.

RESERVA AUTOMÁTICA DE RECURSOS ADMINISTRADOS POR LOS PLANIFICADORES DE RECURSOS. En esta estrategia los recursos son administrados por un sistema local de lotes de colas que se encarga de realizar la reservación y cuanta con un planificador global, el cual co-asigna los recursos distribuidos a cada usuario dependiendo de las necesidades que este demande. Esta estrategia permite una mejor y estructurada administración de recursos ya que se soporta sobre la base de políticas propias para cada organización virtual, evitando los errores de configuración que se presentan en la anterior estrategia.

En consecuencia y con el fin de utilizar eficientemente los recursos en las diferentes políticas de dominio, la mayoría de los recursos de la Grid deben ser gestionados de manera eficiente por los planificadores de recursos. Los planificadores globales tienen que proporcionar recursos para usuarios a nivel mundial en coordinación con los planificadores de los recursos existentes (Takefusa et al., 2007). Aunque existen varios estudios sobre la arquitectura que debería tener el planificador y una amplia discusión sobre si este debería ser centralizado o distribuido (Bolívar y Duran, 2010), debido a las diferentes restricciones de acceso y heterogeneidad en la grid, el presente trabajo centra su investigación en la especificación y complejidad de los algoritmos de búsqueda en un espacio de estados con barrido hacia atrás, con un enfoque de trabajo en varios sub-objetivos independientes, integrando sub-planes a través de la definición de estados, acciones y objetivos (Russell y Norving, 2009), buscando una representación suficientemente expresiva para describir un amplio rango de problemas pero

suficientemente restrictivo para permitir algoritmos eficientes y operativos.

Este enfoque presenta la ventaja de flexibilizar el orden en el que se asignan los recursos a las diferentes tareas a partir de políticas de optimización y priorización.

En el enfoque tradicional donde se busca primero el mejor recurso, de acuerdo a las restricciones de la tarea, una función heurística admisible no se logra por lógica sino por consenso ya que debe satisfacer:

$$f(G_2) = g(G_2) + h(G_2) = g(G_2) > C^*$$

Donde C^* es el costo de la solución optima, G_2 es sub óptimo y $h(G_2) = 0$, cuando se tiene un estado objetivo, pero dicho estado es dependiente de la institución y organización virtual donde se encuentre el recurso, por tal razón para una institución un estado objetivo puede ser incluso despreciable pero para otra puede ser trascendental, fuera de esto la implementación de la búsqueda de estados hacia adelante y hacia atrás en las diferentes simulaciones realizadas, el número de estados dentro de la curva de nivel del objetivo es exponencial, debido al crecimiento del error en la función heurística por encima del logaritmo del coste de la solución optima incumpliendo la condición de crecimiento sub exponencial:

$$|h(n) - h^*(n)| \leq \mathrm{O}(\log h^*(n))$$

Donde $h^*(n)$ es el coste real de alcanzar el objetivo desde n, y para un número de nodos significativos, el consumo de recursos y tiempo en solo la planificación de las tareas obligaría a buscar otras alternativas de procesamiento. Para solventar este problema y buscar una formulación lógica de la función heurística, en primer lugar se busco establecer un lenguaje para la representación de los diferentes problemas que debe afrontar el planificador, para el cual se propuso el *Action Description Language* (ADL) debido a la facilidad de descomponer el problema en condiciones lógicas por medio de la representación de secuencias de literales positivos, enseguida se construyo un algoritmo de planificación prototipo el cual se implemento en C y se ejecuto sobre una simulación en función del número de nodos y las restricciones establecidas, se evidencio un crecimiento de orden $\mathrm{O}(n^2)$ en el algoritmo y en el presente proyecto se busco subdividir el problema y los algoritmos para acercarse a un orden $O(n \log n)$ y finalmente se realizo el análisis de complejidad del algoritmo para contrarrestar los datos obtenidos en las pruebas.

Metodología

En primer lugar se definió la sintaxis para la representación del problema, por medio de la especificación de estados, para lo cual se utilizaron secuencias de literales positivos como condiciones lógicas, primero literales proposicionales: por ejemplo, $nodoLibre \wedge accesible$,

representa el estado de un nodo o recurso que puede ser utilizado para la ejecución de una tarea. También se utilizaron literales de primer orden como por ejemplo,

$En(nodo_1, UCdeC) \wedge En(nodo_{2,} UTP)$,

que representa un estado en el problema de reparto de un trabajo. Luego se realizo la representación sintáctica de objetivos, teniendo en cuenta que un objetivo es un estado parcialmente especificado, dado que teniendo un estado proposicional *s* se satisface un objetivo g, si y solo si *s* contiene todos sus elementos en g (Russell y Norving, 2009). Por ejemplo, el estado $nodoLibre \wedge accesible \wedge En(tarea_1, UCdeC)$ satisface el objetivo $En(tarea_1, UCdeC)$. Luego para la representación sintáctica de las acciones se genero un esquema para cada acción, especificándola junto con su precondición y post condición como se puede observar en la Figura 1, donde se presenta el esquema de acción sobre la carga de una tarea en un nodo.

$Accion\ (tarea\ (x_1, nodo_{local}, nodo_n))$
PRECONDICIÓN: $En(x_1, nodo_{Local}) \wedge tarea(x_1) \wedge UCdeC(nodo_{Local}) \wedge UTP(nodo_n)$
POSTCONDICIÓN: $\sim En(x_1, nodo_{Local}) \wedge En(x_1, nodo_n)$

Figura 1. Representación esquemática de una acción

$< algoritmo - GCL >$ ::= [$< declaraciones >$]|[$< cuerpo >$]
$< declaraciones >$ ::= "*De acuerdo a especificación sintáctica*"
$< cuerpo >$ ::= $< sentencia >$
$< sentencia >$ ::= $< sentencia >, < sentencia >$ |
$< asignación >$
$< condicional >$
$< iteración >$
Skip
abort

Figura 2. Gramática generadora de GCL

Para la definición semántica del problema, se realizo una traslación directa a un conjunto de axiomas de estados sucesivos, utilizando lógica de primer orden, especificando los algoritmos en Guarded Commands Languaje (GCL) (Cardoso, 1991) definidos por la gramática que se encuentra en la Figura 2.

Una vez definida la sintaxis y la semántica de representación del problema, se procedió a definir el algoritmo de planificación con búsqueda en espacio de estados en ADL, para la validación lógica de la especificación del problema y en GCL para el análisis de complejidad. Luego se tuvieron en cuenta diferentes criterios de rendimiento y de optimización para el planificador, debido

a que es un problema multi objetivo en su formulación general se pueden distinguir criterios como el porcentaje de utilización de los recursos en general o de cada recurso, balanceo de carga, el uso del sistema, el tiempo de espera en cola, el tiempo de entrega, el rendimiento acumulado, el tiempo de espera, el tiempo de respuesta, plazos incumplidos, equidad, prioridad de los usuarios, fracaso de los recursos, equilibrio de carga, tiempo total ponderado de realización, retrasos ponderados, número de trabajos retrasados, entre muchos otros (Xhafa y Abraham, 2010). Debido a la alta dependencia del tiempo de ejecución de una tarea respecto a la aplicación a la que pertenece y la máquina donde se está ejecutando, para identificar el tiempo de ejecución Al-Azzoni y Dwon (2009), propone clasificar las tareas en grupos (o clases) con distribuciones idénticas para los tiempos de ejecución, para la presente investigación se especificaron 10 escenarios diferentes de simulación, modificando el número de dominios, el número de nodos por dominio y el número de estados por dominio, tal como se muestra en la tabla 1.

Tabla 1. Escenarios de simulación

Escenarios de simulación	Número de dominios	Número de nodos promedio por dominio	Número de estados promedio por dominio
A	280	12	2
B	86	37	8
C	360	15	3
D	79	34	7
E	480	21	4
F	140	6	3
G	50	21	5
H	500	2	3
I	77	33	7
J	360	15	3

Una vez finalizado el algoritmo base por medio de planificación por regresión debido a que permite considerar las acciones relevantes del cual se puede evidenciar el descubrimiento de recursos en la figura 3. Se realizaron 15 simulaciones por cada escenario para el entrenamiento del modelo, en la figura 4 se presenta el porcentaje ponderado de uso de recursos de cada escenario, en la figura 5 se presenta el porcentaje promedio de entregas exitosas en cada una de las 15 simulaciones y como este porcentaje se incrementa a medida que se realizan nuevas simulaciones, enseguida se modifico el tamaño del problema y se

calculo el numero de iteraciones y secuencias que realizaba cada algoritmo, en la figura 6 se presenta el resultado de las ejecuciones en cada estado, variando el tamaño del problema, los datos se presentan aplicando logaritmo en base dos debido a que el número de iteraciones fue de orden cuadrático y para el peor de los caos se obtuvo un valor de 10^{12}. Finalmente se hizo un análisis buscando cuantificar los recursos abstractos que el algoritmo demanda, midiendo su complejidad, cuya utilidad está directamente relacionada con la posibilidad de estimar aunque sea de manera comparativa las demandas de recursos reales (Bohórquez, 2006). Debido a que un trabajo se subdivide en diferentes tareas es necesario que cada tarea se convierta en un problema de planificación en si, por lo tanto para las n tareas que se tienen se busco resolver la planificación, primero para las *n-1* tareas, aplicando la técnica de dividir y vencerás, aprovechado el paralelo y concurrencia que permite los sistemas Grid. Al aplicar este método el problema cambia a un orden de dominio 2^k y se desarrolla parcialmente la planificación, ejecutándose en un tiempo logarítmico, para luego componerse tarea por tarea, para un tiempo total de ejecución de *O (n log n)*, reduciendo en la simulación el tiempo de ejecución en un 20%, como se puede evidenciar en la figura 7. Aunque se optimizo el algoritmo de planificación propuesto, parcializando en primer orden su ejecución la efectividad de este se mantuvo, y los porcentajes de utilización de los recursos y de entregas exitosas se mantuvieron.

Resultados

$Iniciar: (En(T,G) \wedge En(P,DNS))$

$Objetivo: (En(S(T),C) \wedge (S(T) = \sum_{i=1}^{n} S(t_i)) \wedge (S(t) \equiv true)$

$Accion_1: (\zeta(G(V[i]), \xi(T(V[i]))$

$\boldsymbol{PRECONDICION}: En(T,G)$

$\boldsymbol{POSTCONDICION}: En(T(V[i]), G(V[i]))$

$Accion_2: \lambda(G(V[i]), g_i(V[i]))$

$\boldsymbol{PRECONDICION}: En(T(V[i]), G(V[i]))$

$\boldsymbol{POSTCONDICION}: \bigwedge_{i=1}^{n} g_i(V[i])$

$Accion_3: \zeta(g_i, Q(i))$

$\boldsymbol{PRECONDICION}: \bigwedge_{i=1}^{n} g_i(V[i])$

$\boldsymbol{POSTCONDICION}: En(g_i, Q(i))$

$Accion_4: \xi(Q(DNS), G(DNS))$

$\boldsymbol{PRECONDICION}: En(g, Q)$

$\boldsymbol{POSTCONDICION}: En(Q(DNS), G(DNS))$

$Accion_5: \varphi(T, t_i)$

$\boldsymbol{PRECONDICION}: En(T,G)$

$\boldsymbol{POSTCONDICION}: T = \sum_{i=1}^{n} (t_i)$

$Accion_6: \gamma(g_i, Q(i))$
PRECONDICION: $En(g_i, Q(i))$
POSTCONDICION: $\backsim (En(g_i, Q(i)))$
$Accion_7: \zeta(t_i, g_i)$
PRECONDICION: $En\big(Q(DNS), G(DNS)\big)$
POSTCONDICION: $En(t_i, g_i)$
$Accion_8: \alpha(En(t_i, g_i))$
PRECONDICION: $En(t_i, g_i) \wedge En(t_i) \wedge En(g_i)$
POSTCONDICION: $En(t_i) \wedge En(g_i)$
$Accion_{n-2}: \zeta(En(t_i, g_{i+j}))$
PRECONDICION: $En(t_i, g_i)$
POSTCONDICION: $En(t_i, g_{i+1})$
$Accion_{n-1}: \zeta(S(T), \bigwedge_{i=1}^{n} S(t_i)$
PRECONDICION: $En(\bigwedge_{i=1}^{n}(t_i, g_i))$
POSTCONDICION: $S(T) = \sum_{i=1}^{n} S(t_i) \wedge (S(t) \equiv true)$
$Accion_n: \zeta(S(T), C)$
PRECONDICION: $S(T) = \sum_{i=1}^{n} S(t_i) \wedge (S(t) \equiv true)$
POSTCONDICION: $En(S(T), C)$

Donde: g=RECURSO GRID, T=TRABAJO, G=GRID, T=TAREA, P=SCHEDULER GRID, S=RESULTADO C=CLIENTE GRID, ζ = ASIGNAR, ξ = BUSCAR, λ = COMPARAR, φ = DIVIDIR y α = CONSULTAR.

Figura 3. Especificación del descubrimiento de recursos

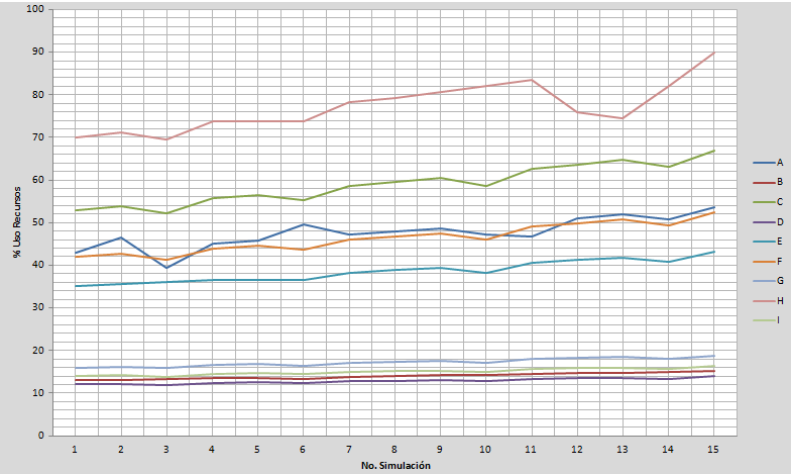

Figura 4. Porcentaje Uso de Recursos de cada escenario

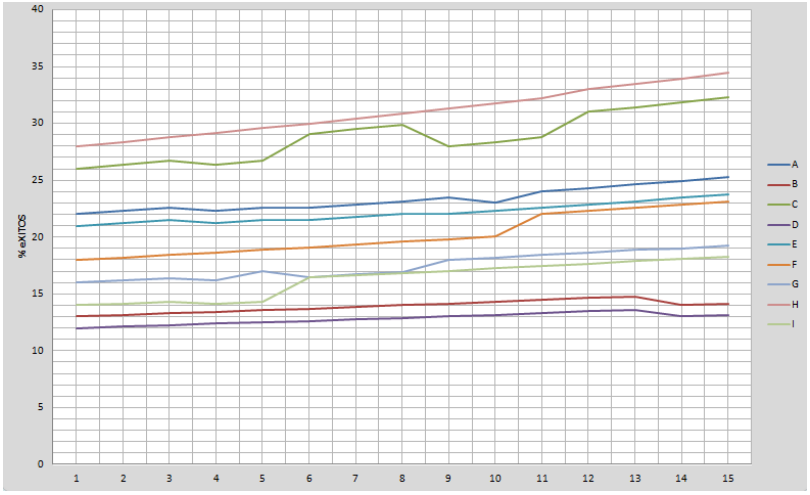

Figura 5. Porcentaje Entregas Exitosas

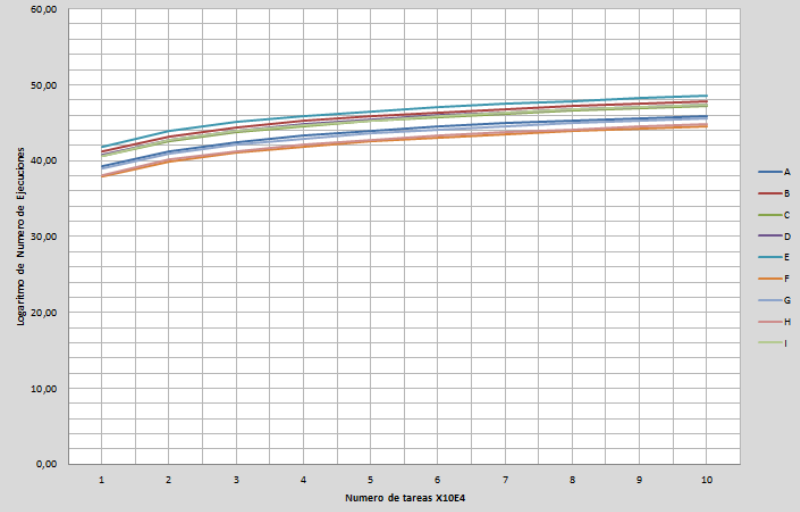

Figura 6. Eficiencia del algoritmo vs tamaño del problema especifico para cada escenario de acuerdo algoritmo base

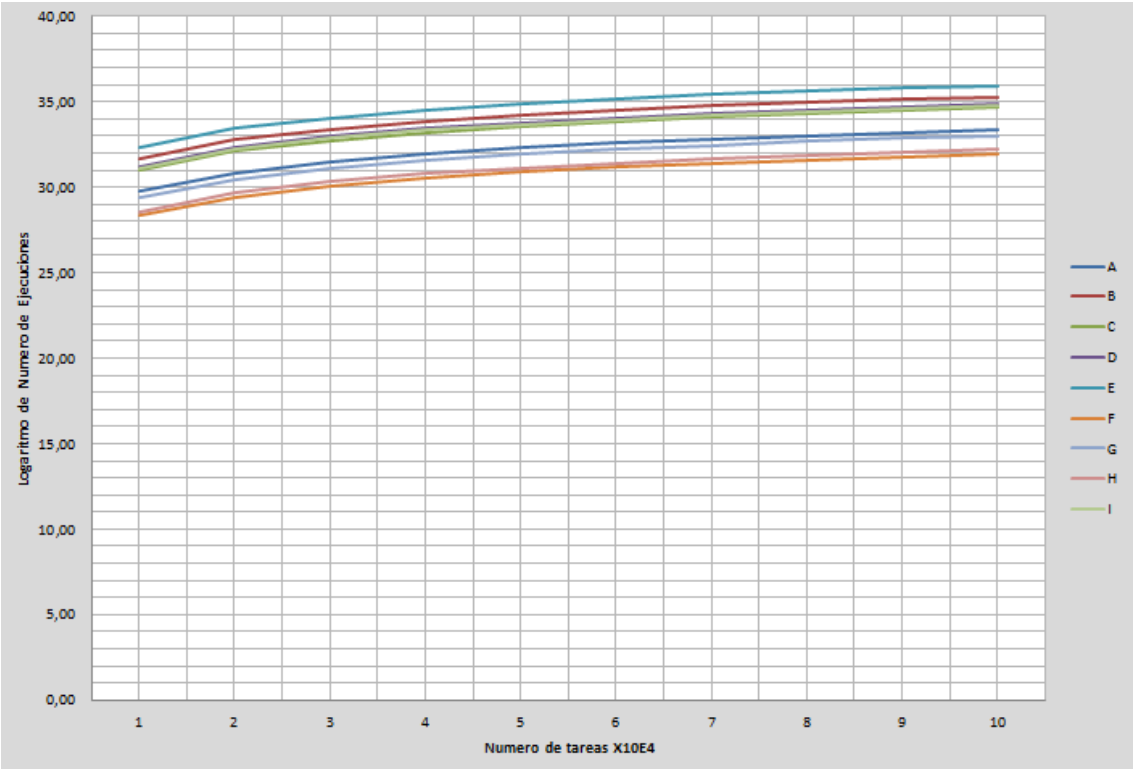

Figura 7. Eficiencia del algoritmo vs tamaño del problema especifico para cada escenario de acuerdo algoritmo óptimo

Discusión

La planificación busca control sobre la explosión combinatoria ya que si se tiene un número *p* de asignaciones de tareas sobre recursos disponibles en la Grid, se puede llegar a tener 2^p estados, al aplicar la técnica de divide y vencerás aplicando el principio de reducción con las interacciones negativas el planificador de orden parcial permite el uso de enlaces causales construyendo sub-objetivos óptimos, eliminando la sobre estimación de búsquedas en múltiples objetivos.

Con la sola eliminación de interacciones negativas no se reduce el problema, ya que se debe especificar las restricciones con que se cuenta en el cumplimiento de cada sub-objetivo, como es el caso de que un nodo o un recurso no esté disponible o sea inaccesible debido a problemas de red, estas validaciones se pueden construir por medios de la construcción de una función heurística basada en el histórico de las asignaciones, lo cual permite el entrenamiento del modelo permitiendo una mayor eficacia en el uso de los recursos y en las entregas de trabajos, de acuerdo a los resultados de la simulación

entre mayor entrenamiento hay, la heurística tendrá mayor información para filtrar asignaciones de recursos que no sean efectivas y esto reduce los tiempos de procesamiento y comparaciones innecesarias.

Analizando el tiempo de ejecución del algoritmo de orden parcial con el esquema propuesto se obtuvo la siguiente recurrencia:

$$t(n) = \begin{Bmatrix} g(n) \\ 2t\left(\frac{n}{2}\right) + f(n) \end{Bmatrix}$$

Donde *g(n)* es el tiempo de generar la solución y *f(n)* el de combinar los resultados, de lo cual se obtiene:

$$t(2^k) = 2t(2^{k-1}) + f(2^k) =$$
$$2\left(2t(2^{k-2}) + f(2^{k-1})\right) + f(2^k) =$$
$$2^m g(2^{k-m}) + \sum_{i=1}^{m-1}\left(2^i f\left(2^{k-i}\right)\right) =$$
$$\theta(n \log_2 n)$$

Conclusiones

La descomposición del problema del planificador en sub-problemas, permite reducir el tiempo teórico de ejecución de $\theta(n^2)$ a $\theta(n \log_2 n)$. El porcentaje ponderado de uso de los recursos del sistema, después de la última simulación para el dominio uniforme fue del 66% con un porcentaje de entregas del 73%, lo cual refleja un incremento ponderado del 8% con respecto a la primera simulación, donde no se tiene entrenamiento sino solo la información del WMS a partir de un GIS centralizado.

Aunque hay incremento y es constante en el tiempo, cada vez es menor y para asegurar un porcentaje de efectividad del 90% se requerirían más de 15000 simulaciones, sobre una misma especificación del sistema, por lo tanto como investigación futura se pretende identificar un modelo para asignar la distribución de probabilidad asociada a cada nodo de trabajo que permita un porcentaje de efectividad mayor en menor tiempo.

Un problema que se deja para una futura investigación es la implementación de esta propuesta en una *grid* real, ya que en la presente investigación se ha realizado por simulación, debido a que una abstracción de alto nivel en la *grid* hace caso omiso de algunos componentes de infraestructura tales como la autenticación, autorización y control de acceso entre otros.

Bibliografía

AL-AZZONI, Issam and DOWN, Douglas. Decentralized load balancing for heterogeneous grids. En: *Future computing, service computation, cognitive, adaptive, content, pattern*. 2009, pp. 545-550. E-ISBN: 978-0-7695-3862-4

BOHÓRQUEZ, Jaime. *Diseño efectivo de programas correctos*. Bogotá: Editorial Escuela Colombiana de Ingeniería, 2006, 50 p.

BOLÍVAR, Holman y DURAN, Edwin. Planificador de tareas multi agente para Grid Colombia integrado con JADE. En: *Studiositas*, 2010, vol. 5, núm. 3, pp. 139 – 158. ISSN 1909-0366.

CARDOSO, Rodrigo. *Verificación y desarrollo de programas*. Bogotá: Ediciones Uniandes, 1991, 36 p.

CHUBURU, Martín, ECHAIZ, Javier, ARDENGHI, Jorge. Servicios de descubrimiento de recursos en sistemas distribuidos de gran escala. En: *Conferencia IADIS Ibero-Americana*. 2008, pp. 257-264.

CHUNLIN, L., ZHENDING, L., LAYUAN, L. Apply Market Mechanism to Agent-Based Grid Resource Management. En: *International Journal of Software Engineering and Knowledge Engineering*, 2003, num. 13, pp. 327-340.

CHUNG, C. Y CHANG, R. A new mechanism for resource monitoring in grid computing. En: *Future Generation Computer Systems*, 2009, núm. 25, pp. 1-7.

CORMEN, Thomas. LEISERSON, Charles, RIVEST, Ronald, STEIN, Clifford. Introduction to algorithms. Massachusetts: The MIT Press. 2001, p. 906-932.

Cox D.P., AL-NASHIF, Y., HARIRI, S. Application of autonomic agents for global information grid management and security. En: *Proceedings of the 2007 summer computer simulation conference*, pp. 1147-1154.

DONG, F. S. y AKL. G. Scheduling algorithms for grid computing: state of the art and open problems. School of Computing. Queen´s University. 2006, p. 55

DONG, Wenli. Grid Analysis Environment Service Architecture for Structured Modeling. En: *2009 E-Learning, E-Business, Enterprise Information Systems, and E-Government*, pp. 53-56.

HAMAR, Vanessa. *Grid computing*. Editorial CICOS. 2009, pp. 144-152

HURTADO, Ricardo y RODAS, Alejandro. Grid: una manera de aparecer en el mundo. En: *Entre Ciencia e Ingeniería*, 2009, No. 5, pp. 130-143

LEAL, Katia, HUEDO, Eduardo y LLORENTE, Ignacio. Dynamic Objective and Advance Scheduling in Federated Grids. En: *Lecture Notes in Computer Science*, 2008, Volume 5331/2008, pp. 711-725. DOI: 10.1007/978-3-540-88871-0_50.

NAKADA, Hidemoto, TAKEFUSA, Atsuko, OOKUBO, Katsuhiko, KISHIMOTO, Makoto, KUDOH, Tomohiro, TANAKA, Yoshio, SEKIGUCHI, Satoshi. Design and implementation of a local scheduling system with advance reservation for co-allocation on the grid. En: *Proceedings of the Sixth IEEE International Conference on Computer and Information Technology*. 2006, 65 p.

RUSSELL, Stuart y NORVING, Peter. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2009, 429 p.

SANG-MIN, Park. y JAI-HOON, Kim. Chameleon: A Resource Scheduler in A Data Grid Environment. En: *IEEE Explore. Digital Library*. 2003, p. 2.

SAPRA, P. y SUNAINA, M. Multi-Agent Systems for Adaptive and Efficient Job Scheduling Service in Grids. En: International Journal of Computer Applications. 2010, núm. 9, pp. 26-30.

SHAN, Hongzhang, OLIKER, Leonid. Job Superscheduler Architecture and Performance in Computational Grid Environments: Super scheduler architecture. En: *Proceedings of the 2003 ACM/IEEE conference on supercomputing*. 2003, p. 44.

TANG, J. y ZHANG, M. An agent based peer to peer grid computing architecture: convergence of grid and peer to peer computing. En: *Proceedings of the 2006 Australasian workshops on Grid computing and e-research*. 2006, vol. 54, pp. 33-39.

TAKEFUSA, Atsuko., NAKADA, Hidemoto, KUDOH, Tomohiro, TANAKA, Yoshio, SEKIGUCHI, Satoshi. Gridars: An Advance Reservation-Based Grid Co-Allocation Framework for Distributed Computing And Network Resources. JSSPP'07 Proceedings of the 13th international conference on Job scheduling strategies for parallel processing. 2007, pp. 152-168.

WALDROP, Mitchell. The Dream Machine: J.C.R. Licklider and the Revolution That Made Computing Personal. Primera Edición. New York. 2001, p. 235.

WU, J., XU, X., ZHANG, P., LIU, C. A novel multi-agent reinforcement learning approach for job scheduling in Grid computing. Future Generation Computer Systems. 2011, núm. 27, p. 430-439.

XHAFA, A., y ABRAHAM, F. Computational models and heuristic methods for Grid scheduling problem. En: *Future Generation Computer System*. 2010, vol. 26, pp 608-621

XINJUN, Chen. WENTONG, Cai. TURNER, Stephen. YONG, Wang. SOAR-DSgrid: service-oriented architecture for distributed simulation on the grid. Principles of Advanced and Distributed Simulation. 2006, p. 65-73.

ZHONGZHI, S., HUANG, H., LUO, J., LIN, F., ZHANG, H. Agent-based grid computing. En: *Applied Mathematical Modelling*, 2006, núm. 30, pp. 629-640.