Sobre la Calendarización en la Grid

Johnatan E. Pecero Sánchez
Instituto Nacional Politécnico de Grenoble
Laboratorio de Informática de Grenoble
Montbonnot Saint-Martin, 38330, Francia
jonathan.pecero-sanchez@imag.fr

Resumen

En este artículo revisamos el problema de la calendarizacón en la Grid. Se hace un repaso principalmente desde el punto de vista de los algoritmos de calendarización. Algunos retos para los tradicionales algoritmos de calendarización frente al problema de calendarización en la Grid son identificados. Se discuten brevemente algunas diferencias entre la calendarización tradicional y la calendarización en la Grid. Entonces, describimos algunas metodologías tradicionales que podrían ser investigadas en el problema de calendarización en la Grid.

Abstract

In this paper, we review the Grid scheduling problem of point of view of traditional scheduling algorithms. In this survey, some challenges for traditional scheduling algorithms when scheduling in Grid computing are identified. We first survey the traditional scheduling problem. After, we focus on Grid computing and Grid scheduling. We discuss briefly difference between traditional scheduling and Grid scheduling. Then, we describe some approaches already adopted in traditional scheduling which could be investigated in Grid scheduling.

1. Introducción

La última década ha visto un substancial incremento en cómodos sistemas de cómputo y eficientes conexiones de red, principalmente como resultado de la rápidez de ejecución en el hardware y del desarrollo de software sofisticado. Estas comodidades tecnológicas se han usado para desarrollar sistemas de cómputo de alto rendimiento (*High-Performance Computing* - HPC systems, en inglés) a un bajo costo, popularmente llamados *clusters* [30], para resolver problemas que demandan el uso de una gran cantidad de recursos y de cálculo intensivo en un gran número de aplicaciones.

Informalmente, un cluster es un sistema de cálculo compuesto por una decena o un centenar de procesadores (típicamente computadoras personales, estaciones de trabajo, etc.) (casi) idénticos (esto es, procesadores homogéneos) conectados conjuntamente a través de una red de alta velocidad y que trabajan como uno solo e integrado sistema de cálculo. Naturalmente, el siguiente paso para resolver la creciente demanda del poder de cálculo de las aplicaciones, es la extensión a conjuntos locales de clusters (cluster de clusters) o la conexión de clusters geográficamente distantes conocido como la tecnología *Grid* computacional [15, 17].

La Grid computacional es una nueva e inovadora tecnología que permite utilizar de forma coordinada todo tipo de recursos (entre ellos cómputo, almacenamiento y aplicaciones específicas) que no están sujetos a un control centralizado. En este sentido, es una nueva forma de computación distribuida en la cual los recursos pueden ser heterogéneos (diferentes arquitecturas, supercomputadores, clusters...) y se encuentran conectados mediante redes de área extensa (por ejemplo Internet) que no son forzosamente rápidas.

Un factor crucial en el uso eficiente y eficaz del poder de cálculo ofrecido por las nuevas tecnologías de cómputo tales como la Grid computacional es la administración de los recursos, principalmente el problema de calendarización de tareas. Este consiste en determinar una ubicación de las tareas de una aplicación a los recursos computacionales y la fecha en la que éstas tareas iniciarán su ejecución, tomando en cuenta criterios de evaluación de *performance* sujeto a la satisfacción de restricciones [23]. Las preguntas naturales e inmediatas que surgen, ¿Es la calendarización en la Grid un nuevo problema el cual difiere de la calendarización tradicional?. No hay una evidencia clara de que éste difiera en una manera fundamental del problema tradicional de calendarización. La diferencia fundamental podría ser la naturaleza dinámica de los recursos y las restricciones en el ambiente de la Grid, pero este punto no es claro aún. ¿Cómo formalizar el problema de la calendarización en la

Grid?. ¿Cómo nos podríamos servir de las metodologías tradicionales de calendarización para resolver el problema de calendarización en la Grid?. En este sentido, sería importante investigar cuáles de las metodologías o algoritmos existentes podrían ser utilizados en este nuevo y complejo problema. Otra pregunta interesante es ¿Cuáles son los retos para las metodologías tradicionales al enfrentarse al problema de la calendarización en la Grid?. En su caso cómo adaptarlos.

En este artículo, estámos interesados en resolver algunas de esas preguntas. Centraremos nuestra atención en el problema de calendarización en los nuevos sistemas de cálculo, específicamente en la Grid desde el punto de vista de la calendarización de tareas. Comenzaremos por revisar el problema clásico de calendarización que ha sido investigado tradicionalmente. Enseguida, nos enfocaremos en la tecnología Grid y el problema de calendarización en la Grid. Más que investigar el problema mismo haremos un repaso del tema, principalmente, desde la perspectiva de algoritmos de calendarización. Después, describimos brevemente algunas de las técnicas tradicionales de calendarización que por su naturaleza misma (la manera como han sido concevidas) podrían ser adaptadas y aplicadas al problema de calendarización en la Grid. Finalmente, terminaremos nuestro trabajo con algunas conclusiones y direcciones futuras.

2. El Problema de Calendarización Tradicional

El problema general de calendarización ha sido descrito de diferentes maneras en la literaura [11, 26]. Usualmente es una redefinición de la noción clásica del problema de *secuenciación de trabajos* [33, 34] en el estudio de la administración de producción.

Desde el punto de vista de calendarización de trabajos distribuidos, Casavant y Kuhl [36] consideran la función de la calendarización como un *recurso para la administración de recursos*. Esta administración de recursos es básicamente un mecanismo o política usada para el manejo eficiente y efectivo del acceso a los recursos y el uso de los mismos recursos por sus varios consumidores. De esta manera, consideran que cada instancia del problema de calendarización consiste de tres componentes principales.

- Consumidor(es)
- Recurso(s)
- Política

Al igual que otros problemas de administración o de control, el entender el funcionamiento de un sistema de calendarización puede hacerse observando el efecto que éste tiene en su ambiente. En este caso, uno puede observar el comportamiento del sistema de calendarización en términos de cómo la *política* afecta los *recursos* y *consumidores*. Aunque haya una sola política, el sistema de calendarización puede ser visto en términos de cómo éste afecta tanto a los recursos como a los consumidores. Esta relación entre el calendarizador, políticas, consumidores y recursos se muestra en la Figura 1.

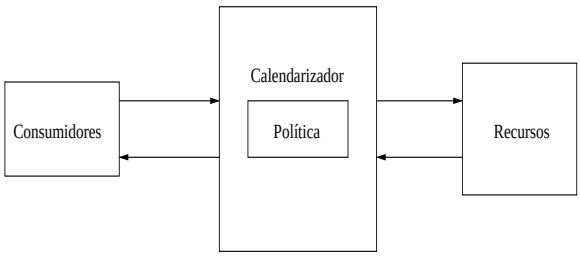

Figure 1. Sistema de Calendarización.

En esta descripción del problema de calendarización, hay dos propiedades las cuales deben ser consideradas en la evaluación de cualquier sistema de calendarización:

1. La satisfacción de los consumidores, en términos de calidad o performance en la administración de los recursos. Es decir, qué tan bien el calendarizador administra los recursos en cuestión, y

2. La satisfacción de los consumidores en términos de eficiencia. Es decir, qué tan difícil o costozo es el acceso a la administración misma de los recursos.

En otras palabras, el consumidor quiere ser capaz de accesar de manera rápida y eficiente los recursos actuales en cuestión, pero no desean tener dificultades por los problemas de sobrecosto asociados con el uso mismo de la función de administración.

Como producto de este enunciado del problema general de calendarización está la unificación de dos términos usados en común en la literatura. Hay una implícita distinción entre los términos de calendarización y ubicación. Estos términos son formulaciones alternativas del mismo problema, con la ubicación desde el punto de vista de los recursos (planteada en términos de la *ubicación de recursos*), y la calendarización desde el punto de vista del consumidor. En este sentido, ubicación y calendarización son méramente dos términos describiendo el mismo mecanismo general, pero utilizados desde diferente punto de vista.

Desde el punto de vista de los recursos, el problema de calendarización trata con la asignación óptima de una aplicación, descrita como un conjunto de tareas, a los elementos de cálculo (procesadores) en un sistema distribuido, de tal manera que el tiempo total de ejecución de las tareas sea minimizado. Uno de los objetivos tradicionalmente investigados es la minimización del tiempo máximo de

terminación sobre todos los procesadores, conocido como *makespan*. Este problema, llamado problema de *calendarización de tareas*, es conocido por pertenecer a la clase de problemas *NP-duro*[1] en su versión más simple, es decir sin tomar en cuenta retardos debido a la comunicación entre las tareas (ver Ullman [21] para mayor refererencia). Como consequencia de la NP-durez del problema de calendarización, muchos métodos han sido desarrollados (con o sin garantías en el resultado) y algunos de ellos han sido implementados en sistemas paralelos tradicionales de la antigua generación de sistemas paralelos [37].

3. La Grid Computacional

La Grid es una nueva tecnología que permite la integración y colaboración de una gran variedad de recursos computacionales, de almacenamiento, heterogéneos, geográficamente distribuídos, administrados por diferentes instituciones (llamadas *organizaciones virtuales*) que ofrecen un gran potencial de cálculo para resolver problemas computacionales de gran escala y que involucran una enorme cantidad de datos en aplicaciones reales [17, 18, 27]. En este ambiente, los recursos computacionales tales como supercomputadoras, clusters, PCs, o estaciones de trabajo, los dispositivos de almacenamiento, software de aplicación, sistemas de bases de datos, etc., son unidos lógicamente y presentados u ofrecidos al usuario como un simple e integrado recurso. Todos estos componentes estan conectados a través de Internet y un software de negociazión que proporcionan servicios básicos para la seguridad, monitoreo y administración de los recursos. Los recursos poseídos por cada organizacion administrativa son compartidos bajo políticas locales definidas que especifican qué se comparte, quién tiene acceso permitido a qué y bajo qué condiciones [16].

La Grid computacional es una infraestructura hardware y software que suministra acceso seguro, consistente, penetrante y económico a las capacidades computacionales [17]. Con respecto a la seguridad en la Grid, ésta está sustentada con las intergrids, donde esa seguridad es la misma que ofrece la red Lan sobre la cual se utiliza tecnología Grid, de tal manera que el acceso sea seguro en todos los niveles: desde capacidad de cómputo hasta la integridad de datos. El servicio debe ser consistente, basado en estándares y de esta manera el acceso y las operaciones sobre la Grid, estarán definidos por dichos estándares evitando la heterogeneidad. La idea de penetración no se refiere tanto a la posibilidad de acceder a cualquier recurso de la Grid, sino que una vez conectado, desde cualquier punto puede extraer de éste toda la potencia que requiera. Por último el acceso y el uso de la Grid debe tener un coste económico que lo haga atractivo para que su utilización sea universal.

Uno de los retos más importantes en la Grid computacional es la eficiente calendarización sobre los recursos distribuidos. La calendarización en el ambiente de la Grid computacional es un problema complejo debido a la gran cantidad de recursos distribuidos geográficamente con diferentes políticas de uso, que pueden exhibir características de rendimiento no uniformes, heterogéneos por naturaleza y con gran disponibilidad.

[1]En teoría de la complejidad computacional, es un problema para la solución del cual no se conoce un algoritmo con complejidad de orden polinomial que lo resuelva.

4. Calendarización en la Grid

La Grid computacional es un término realmente nuevo, con diferentes definiciones en la literatura (ver por ejemplo [15, 17, 18, 27]). En este trabajo adoptamos una definición de la Grid, desde el punto de vista de los sistemas de calendarización, con un cierto grado de abstracción ignorando algunos componentes de la infraestructura tales como la autentificación, autorización, recursos de descubrimiento y control de accesos. De esta manera, la Grid computacional es: *"Un tipo de sistema paralelo y distribuido que permite compartir, seleccionar y agregar recursos geográficamente distribuidos, autónomos y heterogéneos dinámicamente en tiempo de ejecución dependiendo de su disponibilidad, capacidad, performance, costo y la calidad del servicio requerido por el usuario"* [25].

La Grid computacional es una plataforma dinámica, por lo tanto, se debe de tomar en cuenta los parámetros para el sistema de calendarización dinámico, en particular, la estimación del tiempo de ejecución del trabajo, tiempo de llegada y restriccón de precedencia para la calendarización online, además también la cantidad de los recursos computacionales, velocidad efectiva, carga y tipos de tareas. Todos estos parámetros pueden ser desconocidos y variar en el tiempo [9, 35].

4.1. Terminología

La terminología de calendarización usada por la comunidad de la Grid parece un poco diferente de la utilizada por la comunidad de la calendarización tradicional. A continuación se listan algunas definiciones básicas las cuales pueden aclarar o extender la terminología de la calendarización en la Grid.

- Una **tarea** es una operación o unidad atómica para ser ejecutada o procesada por los recursos.

- Las **propiedades** de una tarea son los parámetros como los requerimientos de CPU/memoria, prioridades, fecha máxima de ejecución, etc.

- Un **trabajo** es un conjunto simple de operaciones atómicas que serán realizadas sobre un conjunto de recursos. Los trabajos pueden tener una estructura recursiva, significando que un trabajo puede estar compuesto por sub-trabajos y/o tareas, y los sub-trabajos pueden estar compuestos por mas sub-trabajos y así recursivamente. En este artículo nos referimos a trabajos o aplicaciones indistintamente.

- Un **recurso** es "algo" que es requerido para procesar o ejecutar una operación, como por ejemplo, un procesador para el procesamiento de datos, dispositivos de almacenamiento, etc.

- Un **sitio** o **nodo** es una entidad autónoma compuesta de uno o múltiples recursos.

- La **calendarización de tareas** (desde el punto de vista de la Grid) es similar a la definición utilizada en la calendarización tradicional. Es el mapeo de las tareas hacia un seleccionado grupo de recursos los cuales pueden estar distribuidos en múltiples dominios administrativos, asignando los tiempos de inicio de ejecución de las tareas.

- **Flujo de trabajo** o workflow, éste es la ordenación de trabajos para un usuario específico.

4.2. La Calendarización Tradicional *vs.* la Calendarización en la Grid: Nuevos Retos

El problema de calendarización en la Grid puede ser difinido como: Dado un conjunto de aplicaciones Grid (usualmente cálculos intensivos), cómo calendarizarlos sobre los múltiples recursos decentralizados.

Cada aplicación tiene un número de tareas. En el mapeo de tareas a los recursos, tenemos varias preguntas básicas las cuales tratan con:

1. ¿Cómo la relación entre las tareas afectan la decisión de calendarización?

2. ¿Cómo la heterogeneidad de los recursos afectan el rendimiento de una calendarización?

3. ¿Qué modelo de performance podría usar un sistema de calendarización para determinar la calidad de la calendarización?

Las tareas pueden ser dependientes unas de otras, lo cual hace la calendarización más difícil. De hecho, la mayoría de los trabajos en calendarización han tratado con tareas independientes, divisibles cargas de trabajo, etc. Esas aproximaciones típicamente usan métodos analíticos para hacer calendarizaciones determinísticas o análisis de datos empírico y métodos heurísticos para buscar buenas soluciones.

Algunas de las características principales de los sistemas tradicionales de calendarización son:

- Todos los recursos residen en un solo dominio administrativo.

- Proporcionan una imagen simple del sistema, el sistema de calendarización controla todos los recursos.

- La contención causada por las aplicaciones entrantes pueden ser administradas por el sistema de calendarización de acuerdo a algunas políticas, por lo que su impacto sobre el rendimiento que el sitio pueda proporcionar en cada aplicación puede ser bien predecido.

- Los cálculos y datos residen en el mismo sitio o la organización de los datos es un proceso altamente predecible, usualmente desde una fuente hasta un destino predeterminados, los cuales pueden ser vistos como un sobrecosto constante.

La calendarización en la Grid es intrínsecamente más complicada que la calendarización tradicional debido al manejo de recursos de gran escala a través de ciertos límites de administración. En tal ambiente computacional dinámico y distribuido la disponibilidad de los recursos varía dramáticamente. Por lo que la calendarización se vuelve un reto más difícil. Algunos de estos retos en la concepción de algoritmos de calendarización en la Grid, son [12]:

1. **Heterogeneidad y Autonomía**

   Aunque la heteorgeneidad no es nueva en los algoritmos tradicionales de calendarización inclusive antes de la Grid, este permanece lejos de estar complétamente dirigido hacia la Grid. En la Grid no solo los recursos computacionales o de almacenamiento son heterogéneos sino también la heterogeneidad resulta en capacidades diferentes para el procesamiento de trabajos y el acceso a los datos.

   En la calendarización tradicional, los recursos computacionales son manejados por un solo punto de control. El sistema de calendarización no solo tiene la información completa acerca de la dependencia de las tareas y la utilización de los recursos, sino que también administra la cola de tareas. Esto puede fácilmente predecir el comportamiento de los recursos y estar disponible para asignar tareas a los recursos de acuerdo a un cierto requerimiento de rendimiento.

   En cambio en la Grid, los recursos son usualmente autónomos y el sistema de calendarización no tiene un control total de los recursos. Este no puede violar políticas locales de los recursos, los cuales hacen más difícil para el sistema de calendarización en la Grid el

estimar el costo exacto de ejecutar una tarea en diferentes sitios. La autonomía también resulta en la diversidad en las políticas de manejo de recursos locales y de control de accesos, como por ejemplo, las configuraciones de prioridad para diferentes aplicaciones y los métodos de reservación de recursos.

2. **Dinámismo en el Rendimiento**

   Hacer una calendarización factible usualmente depende de la estimación del rendimiento que el recurso candidato pueda proporcionar, especialmente cuando los algoritmos son estáticos. Los calendarizadores en la Grid trabajan en un ambiente dinámico donde los rendimientos de los recursos disponibles cambian constantemente. El cambio viene desde la autonomía del sitio y la competición entre las aplicaciones por los recursos. Debido a la autonomía de los recursos, usualmente éstos no están dedicados a una aplicación específica. Por ejemplo, un trabajo sometido rémotamente a una computadora de un cluster puede ser interrumpido por un trabajo sometido internamente en el cluster el cual tiene una alta prioridad; nuevos recursos pueden unirse los cuales proporcionan mejores servicios; o algunos recursos pueden volverse indisponibles. El mismo problema ocurre con las redes que conectan los recursos en la Grid: el ancho de banda disponible puede ser afectado por los flujos del tráfico en Internet los cuales no son relevantes para los trabajos de la Grid. Para las aplicaciones de la Grid, éste tipo de contención resulta en una fluctuación en su rendimiento, lo cual hace más difícil el evaluar el rendimiento de la calendarización de la Grid bajo modelos clásicos de rendimiento. Desde el punto de vista de la calendarización de trabajos, la fluctuación en el rendimiento puede ser la característica más importante de la computación Grid comparado con los sistemas tradicionales. Un algoritmo de calendarización factible podría estar disponible para adaptarlo a tal comportamiento dinámico. Algunas otras medidas estan también disponibles para mitigar el impacto de este problema, tales como la negociación *Calidad del Servicio* (Quality of Service - QoS, en inglés), reservación de recursos proporcionada por el sistema administrador de recursos y la re-calendarización.

3. **Separación entre la Selección de Recursos y el Cálculo en los Datos**

   En los sistemas tradicionales, los códigos tradicionales de las aplicaciones y los datos de entradas/salida están usualmente en el mismo sitio, o las fuentes de entrada y las destinaciones de las salidas son determinadas antes de que la aplicación sea sometida. El costo por la organización de los datos puede ser olvidado (considerado como costo de valor zero) o el costo es una constante determinada antes de la ejecución y los algoritmos de calendarización no necesitan considerarlo. Pero en la Grid, la cual está formada por un gran número de sitios computacionales heterogéneos y sitios de almacenamiento conectados vía una red de area extensa, los sitios computacionales son usualmente seleccionados por el sistema de calendarización de la Grid de acuerdo al estado del recurso y a ciertos modelos de rendimiento. Adicionalmente, el ancho de banda de la red en cuestión en la Grid está limitada y compartida a través de un fichero que administra los dominios por lo que la comunicación entre dominios no puede ser desconsiderada. Además muchas aplicaciones incluyen gran cantidad de datos, por lo que el costo de la organización de datos es considerable. Esta situación nos lleva al problema de separación datos-computación: la ventaja es que al seleccionar un recurso computacional que puede proporcionar bajo costo computacional puede ser neutralizado por su alto costo de acceso al sitio de almacenamiento.

   Estos retos definen características únicas de la Grid computacional y ponen muchos obstáculos para el diseño e implementación de eficientes y efectivos sistemas de calendarización en tal arquitectura. Parece difícil el panorama para las tradicionales metodologías de calendarización frente a este gran problema. Sin embargo, se cree que los logros obtenidos en los problemas tradicionales de calendarización puedan servir de base cuando una nueva generación de sistemas de calendarización sean construídos [1]. En la siguiente sección presentamos algunas metodologías tradicionales que pueden servir como herramienta en la construcción de eficientes sistemas de calendarización en la Grid.

5. Metodologías Tradicionales de Calendarización

No obstante los retos de la calendarización en la Grid, ésta podría benefisiarse de tradicionales metodologías de calendarización que por su propia naturaleza podrían trabajar bien en el ambiente dinámico y heterogéneo de la Grid computacional. Estas metodologías han logrado resultados acertados e interesantes en una gran variedad de aplicaciones de calendarización. Por lo tanto, resulta interesante el investigar su funcionamiento y rendimiento en los problemas de calendarización en los nuevos sistemas de cálculo. A continuación se describen estas metodologías tradicionales:

5.1. Heurísticas:

Una heurística es una técnica que busca buenas soluciones en un tiempo razonable de cálculo sin poder garantizar su factibilidad u optimalidad, o inclusive en muchos casos sin poder declarar que tan cerca una solución particular está de la solución optimal [3]. Las reglas de prioridad (Dispatching rules, en inglés), son ejemplos de heurísticas. Son usadas para seleccionar el siguiente trabajo para procesar en los recursos cuando un recurso está disponible. Las reglas de prioridad incluyen entre otras, la fecha de entrega más próxima (Earliest Due Date-EDD, en inglés), primero en llegar, pimero en ser atendido (First Come First Served-FCFS), menor tiempo de procesamiento (Shortest Processing Time-SPT), etc.

Dos de los algoritmos más utilizados para la calendarización de tareas basados en heurísticas, son los algoritmos de lista [32] y los algoritmos de agrupamiento o de clustering [38, 39].

La idea de los algoritmos de lista se basa en mantener una lista de prioridad entre las tareas. El principio de dichos algoritmos es de no dejar ningún procesador desatendido si existe alguna tarea que esté disponible y pueda ser executada en ese instante sobre dicho procesador. Estos algoritmos son simples y glotones, aún así han sido utilizados por muchos investigadores obteniendo resultados teóricos interesantes como por ejemplo, el poder garantizar la calidad de la solución obtenida [32]. La diferencia principal entre un algoritmo de lista y otro, se basa en la heurística utilizada para dar prioridad a las tareas, como por ejemplo, dar prioridad a las tareas que pertenecen al camino más largo[2], dar prioridad a las tareas con mayor tiempo de procesamiento, a las de menor tiempo de procesamiento, las tareas más conectadas (con el mayor número de sucesores y predecesores), etc.

Por su parte, los algoritmos de clustering están basados en la agrupación de tareas simples para formar tareas de mayor peso (granularidad más grande [38]), de tal manera que exista un balance entre el tiempo de comunicación y tiempo de ejecución de las tareas. Estos algoritmos, resuelven el problema de calendarización en dos etápas. En la primera, las tareas son agrupadas para formar los clusters considerando una cantidad ilimitada de recursos. En esta primera etapa, las tareas en el interior de un mismo cluster serán ejecutadas en secuencial. Cada cluster se considera como un procesador (procesadores virtuales). En la siguiente etapa, los clusters resultantes son combinados entre sí, de forma tal que el número de clusters sea igual o menor al número de procesadores reales disponibles. Diferentes algoritmos de clustering han sido propuestos en la literatura la diferencia entre uno y otro se basa principalmente en la heurística utilizada para formar los clusters o grupos de tareas como por ejemplo, agrupar las tareas que pertenecen al camino más largo, basado en listas, formar grupos de tareas con cierta estructura.

[2]llamado también, en la gran mayoría de los artículos sobre calendarización, como la ruta crítica

5.2. Metaheurísticas: búsqueda tabú, recocido simulado, algoritmos genéticos:

Las metaheurísticas, son heurísticas de alto nivel que guían heurísticas de búsqueda local para escapar de locales óptimos. En [19], Osman proporciona la siguiente definición de metaheurísticas: *"Dentro de la clase denominada metaheurísticas se incluyen todos aquellos procedimientos que en un proceso iterativo, guían a una heurística subordinada combinando inteligentemente diferentes conceptos tomados de analogías de la naturaleza, y exploran el espacio de soluciones utilizando estrategias de aprendizaje para estructurar la información, con el objeto de encontrar eficientemente soluciones cercanas al óptimo."*

Las metaheurísticas tales como búsqueda tabú, recocido simulado y algoritmos genéticos mejoran los algoritmos de búsqueda local para escapar de óptimos locales aceptando peores soluciones o generando buenas soluciones iniciales de una manera más inteligente que solo generar soluciones iniciales aleatorias [2, 3, 4].

5.3. Sistemas basados en conocimiento:

Sistemas basados en conociemiento enfocados a la captura de la experiencia o del experto en calendarización. Un mecanismo de inferencia es usado para derivar conclusiones o recomendaciones con respecto al problema de calendarización [10, 40].

5.4. Calendarización dinámica

La calendarización dinámica se refiere al problema de calendarización en ambientes dinámicos. La calendarización en la Grid opera en ambientes dinámicos sujetos a varios eventos desconocidos y no planeados que pueden ocurrir. Tales eventos incluyen la caída de algún recurso de cómputo, la llegada de nuevos trabajos, etc. En este caso, el rendimiento de la calendarización es muy sensible a estas perturbaciones, y es difícil ejecutar una calendarización predictiva generada por adelantado. Esos eventos en tiempo real no solo interrumpen la operación del sistema, sino que también afectan la predicción de la calendarización establecida con anterioridad. Consecuentemente el resultado de la calendarización puede no ser factible o no ser el óptimo. La calendarización dinámica es argumentable de importancia práctica en la calendarización en la Grid para generar calendarizaciones robustas. Para mayor referencia, ver [14, 29].

5.5. Lógica difusa:

Los sistemas de lógica difusa o borrosa consisten en una variedad de conceptos y técnicas para representar e inferir conocimiento que es impreciso, incierto, difuso, vago. Es la lógica que utiliza expresiones que no son ni totalmente ciertas ni completamente falsas, es decir, es la lógica aplicada a conceptos que pueden tomar un valor cualquiera de veracidad dentro de un conjunto de valores que oscilan entre dos extremos, la verdad absoluta y la falsedad total. Los modelos de calendarización basados en métodos difusos han atraído reciéntemente el interés entre la comunidad científica de calendarización [31].

Trabajos previos han investigado la representación de incertitud en el tiempo de procesamiento y tiempo previsto a través de la noción de números borrosos (para mayor información sobre esta noción, referimos al lector a [8]), la representación de restricciones flexibles a través de medidas difusas, la relación de precedencia de trabajos o la interrupción de máquinas, pero han sido trabajos aislados. Una vez que los acuerdos de los niveles de servicio han sido acordados, hay muchas maneras en las cuales puede ser necesario la renegociación. En un ambiente Grid completamente ocupado, los acuerdos de niveles de servicio podrían ser agregados, alterados o retirados, y de ahí la calendarización podría necesitar ser un proceso contínuo, dinámico e incierto. Recientes trabajos de investigación han iniciado la utilización de técnicas de lógica difusa en la calendarización en la Grid [13, 22].

5.6. Agentes y Sistemas Multiagentes:

Reciéntemente, los sistemas multiagentes son una de las técnicas más prometedoras para construir una nueva generación de sistemas manufactureros de calendarización complejos, robustos y de bajo costo debido a su autónoma, distribuída y dinámica naturaleza y a su robustéz a las fallas.

Un agente es un sistema de computadora que está situado en algún ambiente y que es capaz de actuar de manera flexible y autónoma en este ambiente para conocer los objetivos diseñados. Ser flexible, significa que el sistema debe ser sensible, capaz de tomar la iniciativa, de ser activo y sociable. Un agente autónomo es un sistema situado que siente ese entorno y actúa sobre él, a través del tiempo, persiguiendo sus propios objetivos de forma que afecte lo que siente en el futuro [24].

Un sistema multiagente es un sistema compuesto de una población de agentes autónomos, los cuales interactúan entre ellos para lograr objetivos comunes, mientras simultáneamente cada agente persigue sus objetivos individuales [7].

Algunas motivaciones para incrementar el interés en la investigación de la calendarización basado en sistemas multiagentes son las siguientes:

- Problemas de calendarización en la vida real están comúnmente distribuidos físicamente o funcionalmente, ejemplos, control de tráfico aéreo, sistemas manufactureros, etc.

- Los sistemas complejos de calendarización van más allá del control directo. Operan a través de la cooperación e interacción de múltiples subsistemas, los cuales pueden tener su propio interés y modos de operación.

- Hay la necesidad de la integración de múltiples sistemas antiguos y expertos.

- Los problemas reales de calendarización son heterogéneos. Ambientes heterogéneos pueden utilizar datos y modelos diferentes, y operar en diferentes modelos.

- Los agentes proporcionan robustéz y fiabilidad a contra fallas. Los sistemas distribuidos permiten una rápida detección y recuperación de la falla de uno o varios agentes no hacen necesariamente inútil el sistema.

- Los agentes pueden operar asíncrona y paralelamente, lo cual puede resultar en un aumento en la velocidad.

Los agentes han sido usados satisfactoriamente para resolver una gran variedad de problemas complejos de calendarización[28, 5, 6]. Por lo tanto, es posible que los sistemas multiagentes proporcionen buenas bases para la creación de sistemas de calendarización en la Grid que posean características de autonomía, heterogeneidad, flexibilidad y robustez. Algunas investigaciones se han iniciado sobre el uso de agentes en la tecnología Grid, en particular para la administración de recursos en el ambiente Grid [20].

6. Conclusiones y Direcciones Futuras

En este artículo se hizo una revisión del estado del arte sobre la calendarización en los nuevos sistemas paralelos y distribuídos. Específicamente la calendarización de aplicaciones en la Grid. En una primera parte, se presentó una visión general del problema de calendarización. En una segunda parte, se presentó de manera general la tecnología Grid y el problema de calendarización en la Grid. Se hizo una analogía entre el problema clásico de calendarización y el complejo problema de la calendarización en la Grid. Se discutió brevemente cómo nos podríamos servir de técnicas tradicionales de calendarización para tratar de resolver los

actuales problemas de calendarización en las nuevas tecnologías de cómputo. Es decir, se describieron algunos retos que deben enfrentar las técnicas tradicionales de calendarización frente a este problema en la Grid. En base a esto, se presentaron algunas estrategias, utilizadas en la literatura para resolver problemas clásicos de calendarización, que por su naturaleza misma ofrecen características bien adaptadas a la Grid computacional.

References

[1] A. Andrieux, D. Berry, J. Garibaldi, S. Jarvis, J. MacLaren, D. Ouelhadj, D. Snelling. Open issues in Grid scheduling. Technical Report UKeS-2004-03, Edinburg, Uk, Oct. 2003. Official Technical Report of the Open Issues in Grid Scheduling Workshop.

[2] C. Blum, A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computer Surveys*, 35(3):268–308, 2003.

[3] C. R. Reeves, (ed.). *Modern heuristic techniques for combinatorial problems*. John Wiley & Sons, McGraw-Hill International (UK) Limited, 1995.

[4] C.R. Reeves, J.E. Rowe. *Genetic Algorithms–principles and perspectives*. Kuwler Academic Publishers, Norwell, MA, USA, 2002.

[5] D. Ouelhadj, C. Hanachi, B. Bouzouia. Multi-agent system for dynamic scheduling and control in manufacturing cells. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1256–1262, Belgium, 1998.

[6] D. Ouelhadj, C. Hanachi, B. Bouzouia. Multi-agent architecture for distributed monitoring in flexible manufacturing systems (fms). In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1120–1126, San Francisco, 2000.

[7] E. Oliveira, K. Fisher, O. Stepankova. Multi-agent systems: which research for which applications. *Robotics and Autonomous Systems*, 7(1-2):91–106, 1998.

[8] E. Ruspini, P. Bonissone, W. Pedrycz (eds.). *Handbook of fuzzy computation*. Institute of Physics Publishing and Oxford University Press, Bristol and Philadelphia, 1998.

[9] E. Salazar, M. Valenzuela, D. Mendoza, Y. Castro, J.M. Ramírez, E. Rodríguez, A. Tchernykh. Evaluación de las estrategias de calendarización en un Grid computacional jerárquico. In *las memorias de la XVI Escuela Nacional de Optimizacón y Análisis Numérico, ENOAN 06*, Tlaxcala, México, Marzo 2006.

[10] E. Szelke, R.M. Kerr, editor. *Knowledge-Based Reactive Scheduling, Proceedings of the IFIP TC5/WG5.7 International Workshop on Knowledge-Based Reactive Scheduling, Athens, Greec, 1 October, 1993*, volume B-15 of *IFIP Transactions*. North-Holland, 1994.

[11] E.G. Coffman, P.J. Denning. *Operating Systems Theory*. Prentice Hall Professional Technical Reference, Englewood Cliffs, N.J., 1973.

[12] F. Dong, S.G. Akl. Scheduling algorithms for Grid computing: State of the art and open problems. Technical Report 2006-504, School of Computing, Queen's University, Jan. 2006.

[13] F. Vraalsen, R.A. Aydt, C.L. Mendes, D.A. Reed. Performance contracts: predicting and monitoring grid application behavior. In *Proceedings of the 2nd International Workshop on Grid Computing (Grid 2001)*, volume 2242 of *LNCS*, pages 154–165. Lecture Notes in Computer Science, Springer-Verlag, 2001.

[14] G.E. Vieira, J.W. Hermann, E. Lin. Rescheduling manufacturing systems: a framework of strategies, policies and methods. *Journal of Scheduling*, 6(1):36–92, Feb. 2003.

[15] I. Foster. What is the Grid: A three point checklist. *In GRIDToday*, July 2002.

[16] I. Foster, A. Iamnitchi. On death, taxes, and the convergence of Peer-to-Peer and Grid computing. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, Berkeley, CA, USA, Feb. 2003.

[17] I. Foster and C. Kesselman. *The Grid: Blueprint for a future computing infrastructure*. Morgan Kaufmann Publishers, USA, 1999.

[18] I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of Supercomputer Applications*, 15(3), 2001.

[19] I.H. Osman, G. Laporte. Metaheuristics: a bibliography. *Annals of Operations Research*, 63(5):513–628, 1996.

[20] J. Cao, D. Spooner, G. Nudd. ARMS: an agent-based resource management system for grid computing. *Scientific Programming*, 10(2):135–148, 2002.

[21] J. D. Ullman. NP-complete scheduling problems. *Journal of Computer and System Sciences*, 10:384–393, 1975.

[22] J. Huang, H. Jin, X. Xie, Q. Zhang. An approach to grid scheduling optimization based on fuzzy association rule mining. In *Proceedings of the First International Conference on e-Science and Grid Computing*, pages 189–195, Washington, DC, USA, 2005. IEEE Computer Society.

[23] M. Pinedo. *Scheduling theory, algorithms and systems*. Prentice Hall, Englewood Cliffs, NJ, 1995.

[24] M. Wooldrige, N.R. Jennings. Intelligent agents: theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[25] M.Baker, R. Buyya, D. Laforenza. Grids and Grid technologies for wide-area distributed computing. *Journal of Software-Practice & Expirience*, 32(15):1437–1466, Dec. 2002.

[26] M.J. Gonzalez. Deterministic Processor Scheduling. *ACM Computing Surveys*, 9(3):173–204, Sept. 1977.

[27] P. Asadzadeh, R. Buyya, C.L. Kei, D. Nayar, S. Venugopal. *High Performance Computing: Paradigm and Infrastructure*. Wiley Press, NJ, USA, June 2005.

[28] P.I. Cowling, D. Ouelhadj, S. Petrovic. A multi-agent architecture for dynamic scheduling of steel hot rolling. *Journal of Intelligent Manufacturing*, 14(5):457–470, 2003.

[29] P.I. Cowling, M. Johansson. Using real time information for effective dynamic scheduling. *European Journal of Operation Research*, 139:230–244, 2.

[30] R. Buyya, (ed.). *High performance cluster computing: architectures and systems*, volume 1. Prentice Hall, NJ, USA, 1999.

[31] R. Slowinski, M. Hapke (eds.). *Scheduling under fuzziness*. Physica-Verlag, New York, USA, 2000.

[32] R.L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Tech. J.*, 45:1563–1581, 1966.

[33] R.W. Conway, W.L. Maxwell, L.W. Miller. *Theory of Scheduling*. Addison-Wesley, Massachussetts, 1967.

[34] S. French. *Sequencing and Scheduling*. John Wiley & Sons, New York, 1982.

[35] S.S. Vadhiyar, J.J. Dongarra. A metascheduler for the Grid. In *Proceedings of the 11th IEEE Symposium on High Performance Distributed Computing (HPDC 2002)*, pages 343–351, Edinburgh, Scotland, July 2002.

[36] T. Casavant, J. Kuhl. A taxonomy of scheduling in general-purpose distributed computing systems. *IEEE Transactions on Software Engineering*, 14(2):141–154, Feb. 1988.

[37] T. Yang, A. Gerasoulis. PYRROS: static task scheduling and code generation for message passing multiprocessors. In *Proceedings of the 6th ACM International Conference on Supercomputing*, pages 428–437, Washington D.C., July 1992.

[38] T. Yang, A. Gerasoulis. DSC: Scheduling parallel tasks on an unbounded number of processors. *IEEE Transactions on Parallel and Distributed Systems*, 5(9):951–967, Sept. 1994.

[39] V. Sarkar. *Partitioning, Scheduling Parallel Programs for multiprocessors*. Pithman, The MIT Press, Cambridge, MA, USA, 1989.

[40] V.C.S Wiers. A review of the applicability of OR and AI scheduling techniques in practice. *Omega International Journal of Management Science*, 25(2):145–153, 1997.