Cátedra Nissan

-PROTHIUS-

Programación de la producción en una empresa alimentaria, utilizando la paralelización en los cálculos

Joaquín Bautista Valhondo, Ramón Companys Pascual, Albert Corominas Subias, Manel Mateo, Rafael Pastor y Eugeni Roures

WP-16/2010
(Rec. DIT 98/12- BCCMPR- 1998)

Departamento de Organización de Empresas

Universidad Politécnica de Cataluña

Publica:

Universitat Politècnica de Catalunya
www.upc.edu

Edita:

Cátedra Nissan
www.nissanchair.com
director@nissanchair.com

TÍTOL:

PROGRAMACIÓN DE LA PRODUCCIÓN EN UNA EMPRESA ALIMENTARIA, UTILIZANDO LA PARALELIZACIÓN EN LOS CÁLCULOS.

Autors:

Joaquín Bautista
Ramon Companys
Albert Corominas
Manel Mateo
Rafael Pastor
Eugeni Roures

Document Intern de Treball (D.I.T.) 98/12

Barcelona, 1 d'abril de 1998

Treball realitzat en el marc del projecte europeu EP-9602.

Programación de la producción en una empresa alimentaria, utilizando la paralelización en los cálculos.

J. Bautista (1), R. Companys (1), A. Corominas (1), M. Mateo (1), R. Pastor (1), E. Roures (2).
(1) Departamento de Organización de Empresas. Universitat Politècnica de Catalunya. Barcelona.
(2) Neosystems Informática S.A. Barcelona.

Resumen:

Una empresa del sector cárnico, que produce bajo pedido, disponía de un algoritmo para la programación de la producción con unos tiempos de explotación que no facilitaban su uso frecuente. La complejidad y variedad de los recursos y los procesos (máquinas polivalentes, máquinas de fabricación continua, recintos de almacenamiento compartidos, rutas alternativas, etc.), y el carácter perecedero del material utilizado, aumentaba la dificultad de alcanzar una respuesta satisfactoria en un tiempo aceptable y obligaban a un trato específico con un nuevo algoritmo.

Con el fin de reducir los tiempos de cálculo, se consideró el uso de paralelismo en aquellos cálculos necesarios para evaluar la asignación de una operación a cada uno de los recursos productivos disponibles. El algoritmo, diseñado pues para calcular en paralelo la asignación de operaciones a los diferentes recursos productivos, se ha desarrollado en el marco de los proyectos PACOS (*Parallel Computing for Spain*).

Palabras clave: programación de la producción, cálculo paralelo, PVM.

ÍNDICE

1. Introducción 2
2. El sistema productivo 3
3. El algoritmo de carga de máquinas 6
4. El nuevo algoritmo en paralelo 8
5. Resultados del nuevo algoritmo 11
6. Conclusiones 15
Bibliografía 15

1. Introducción

En este artículo se expone una aplicación práctica en el campo de la planificación de la producción en una planta de la compañía Hesperia de Alimentación S.A., empresa alimentaria en el sector cárnico, que produce toda clase de productos derivados de la carne (jamones, salchichas, fiambres, etc.). El proyecto ha involucrado a cuatro socios: el usuario final, Hesperia de Alimentación S.A.; una compañía dedicada a *software*, Neosystems Informática S.A.; y dos departamentos universitarios, el Departamento de Estadística e Investigación Operativa y el Departamento de Organización de Empresas, ambos pertenecientes a la Universitat Politècnica de Catalunya. Esta colaboración se ha llevado a cabo en el marco de un proyecto ESPRIT, llamado Sirius, impulsado por la Comunidad Europea e incluido en el programa PACOS (*Parallel Computing for Spain*).

El objetivo principal de este proyecto se centraba en la mejora del entonces vigente algoritmo de carga de máquinas, con el fin de reducir los tiempos de ejecución. Una opción planteada para alcanzarlo consistía en la adaptación del algoritmo secuencial a una nueva versión que realizase los cálculos en paralelo. No obstante, otros objetivos entroncados a éste eran establecer unas reglas de prioridad en el lanzamiento y la secuenciación de las operaciones, independizar el algoritmo de la plataforma de trabajo en que fuera ejecutado y convivializar la interficie de usuario con una adaptación a las nuevas técnicas de representación gráfica en pantalla.

El objetivo básico derivaba de la cuestión a que los planificadores de esta empresa se enfrentaban semanalmente: determinar qué tareas debían realizarse en cada uno de los recursos productivos de que disponían en su fábrica y cuándo. Para ello, se planteaban cómo realizar los productos requeridos a mínimo coste, es decir, utilizando los recursos de la manera más eficiente en el lanzamiento de órdenes de fabricación, que son descompuestas en un conjunto de actividades. La situación tratada puede situarse en el campo de los problemas de "*job-shop*", aunque con diversos matices respecto al caso general, que incrementan la dificultad de tratamiento.

Basándose en anteriores trabajos realizados por algunos de los autores en el campo de la secuenciación de órdenes de producción y en la programación de sus actividades, y adaptando las características de resolución de los problemas "*job-shop*" [2], se ha demostrado cómo los problemas de programación de operaciones en este tipo de industrias se pueden abordar con resultados satisfactorios utilizando algún procedimiento heurístico.

El artículo se organiza tal como se indica a continuación. En la sección 2 se describen las características del sistema productivo en cuestión. En la sección 3 se presenta el algoritmo utilizado en la programación de la producción, y en la siguiente sección, se explica la incorporación del paralelismo en el algoritmo y sus consecuencias. Ya en la sección 5, se facilitan ciertos resultados de la aplicación del nuevo algoritmo sobre datos reales. Finalmente, en la sección 6 se expresan algunas conclusiones obtenidas a partir de este proyecto.

2. El sistema de producción

Ante todo, se ofrece una descripción del sistema de producción sobre el cual debe realizarse la programación con tal de comprender correctamente las particularidades que son tratadas por el algoritmo implementado.

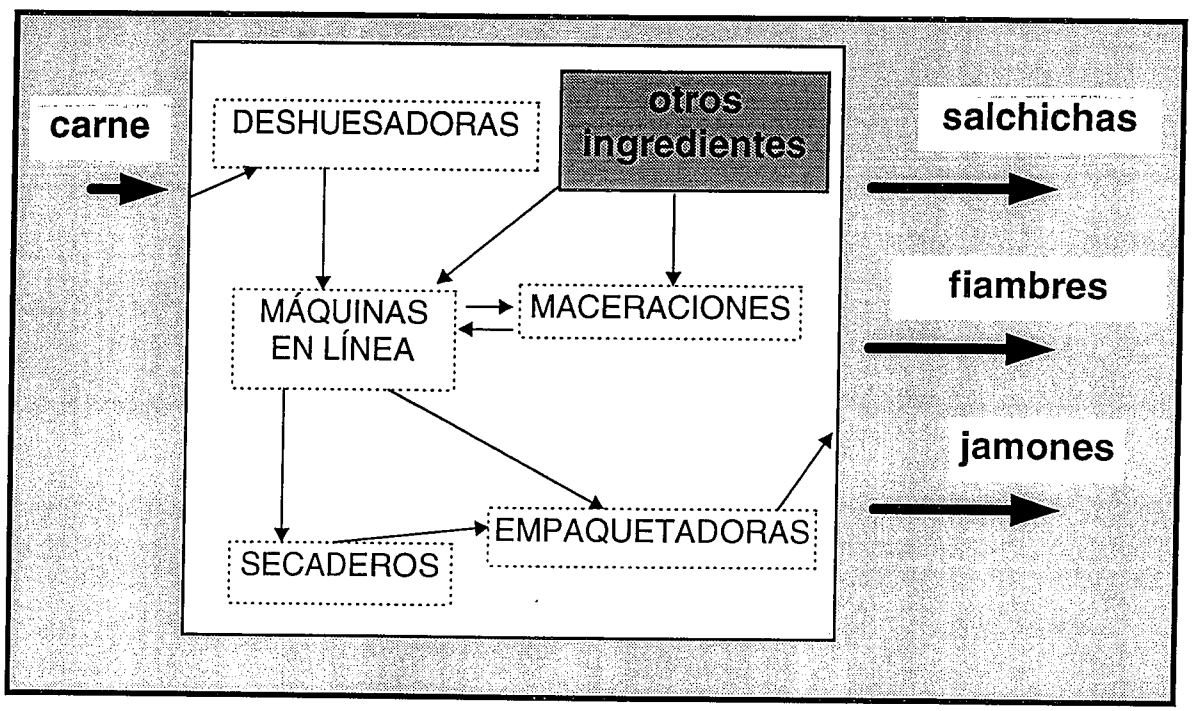
La fábrica se ubica en Cataluña, España, y produce una gran variedad de productos derivados de la carne: jamones, salchichas, etc. Su planta de producción se compone de diversas líneas, cada una de ellas especializada en algunos de los productos de acuerdo con las respectivas características, aunque algunas de las líneas utilizan recursos comunes, lo que obliga a considerar todos ellos a la vez en el proceso de programación.

Los **recursos productivos** considerados son alrededor de 130. El algoritmo se ha desarrollado con el fin de optimizar no tan sólo el uso de las máquinas, los únicos recursos que actualmente se tratan, sino también la cantidad de personal requerido. La disponibilidad de cada uno de los recursos se controla mediante "calendarios", que indican los períodos de fabricación no permitidos para cada recurso.

Los recursos productivos requeridos para realizar las operaciones de transformación de la carne y los demás ingredientes en los productos finales pueden agruparse básicamente en diversos tipos:

- los primeros, son aquellos recursos propios de la planta y capaces de realizar una única actividad;
- otro grupo incluye los recursos "compartidos", que son capaces de realizar más de una de actividad simultáneamente;
- y finalmente, los recursos productivos externos, que se hallan físicamente fuera de la planta.

Por supuesto, la asignación de las actividades simultáneas en un recurso compartido requiere una total compatibilidad entre todas ellas.



Cada producto sigue un cierto proceso productivo. Los **procesos de producción** definidos hasta este momento son alrededor de 30, aunque este valor puede variar según la incorporación de nuevos productos y la supresión de productos que ya no se fabriquen. Las líneas de producción en esta fábrica, como en otras, introducen relaciones de precedencia entre actividades, pero como existen recursos con capacidad muy limitada y algunas actividades requieren una larga duración, estas actividades deben tener lugar simultáneamente en varios recursos de producción. La división de la cantidad total a tratar de un producto en diversas cantidades menores para poder realizar a la vez una actividad en diversos recursos se denomina en esta planta como modularidad.

Los procesos productivos de esta fábrica presentan una diversidad en el número de actividades requeridas para la elaboración de cada producto (desde pocas actividades, cuatro o cinco, hasta unas veinte) debido a la gran variedad de productos fabricables.

Una tarea compleja es el reparto de la cantidad a producir en algunas operaciones modulares en diversas actividades modulares. Véase el ejemplo de la cocción de 2.000 kg al final de este punto.

Presentados los recursos disponibles y el tratamiento dado a la transformación de la carne y las demás materias primas en los productos finales, pasamos a analizar el trato dado a las **órdenes de producción**. El número de órdenes que esta fábrica produce actualmente es de unas 80 por semana, aunque se espera que se incremente debido a un uso eficiente de los recursos, como resultado de esta nueva manera de programar.

El tratamiento dado a las órdenes consiste, conocido qué producto tiene que ser fabricado, en la asignación automática a la orden del proceso de producción que ya está predefinido. Por tanto, las actividades necesarias para realizar la orden se enumeran siguiendo el mismo modelo que el respectivo proceso, adaptando las necesidades de tiempo y cantidades a la cantidad total especificada en esta orden particular.

Las órdenes de fabricación se descomponen en **actividades**, componente básico de éstas y que pueden ser de diversa índole: pueden dividirse en "dinámicas" y "estáticas" o modulares. Aunque estas palabras no se refieren a la duración de las actividades, la principal propiedad que permite establecer la división, son términos usados en la fábrica, asociados a la obra en curso: si el material realiza una operación en movimiento o permanece fijo durante ésta. En las actividades dinámicas, la duración de la actividad es directamente proporcional a la cantidad a ser fabricada. En cambio, en las actividades estáticas o modulares, que pueden realizarse en varios recursos llamados **módulos**, la duración de la actividad depende básicamente de la capacidad física y de la cantidad aún disponible para fabricación en cada recurso.

Una actividad perteneciente a la lista puede asignarse habitualmente a varios recursos de la planta, todos ellos con una determinada función en el sistema productivo. El algoritmo trabaja con **grupos productivos**, que son listas de recursos productivos agrupados de acuerdo con las funciones que puedan desempeñar. Cada una de las operaciones se encuentra relacionada con uno de estos grupos de recursos disponibles en la fábrica, de entre los cuales debe escogerse uno, al cual se asignará dicha actividad.

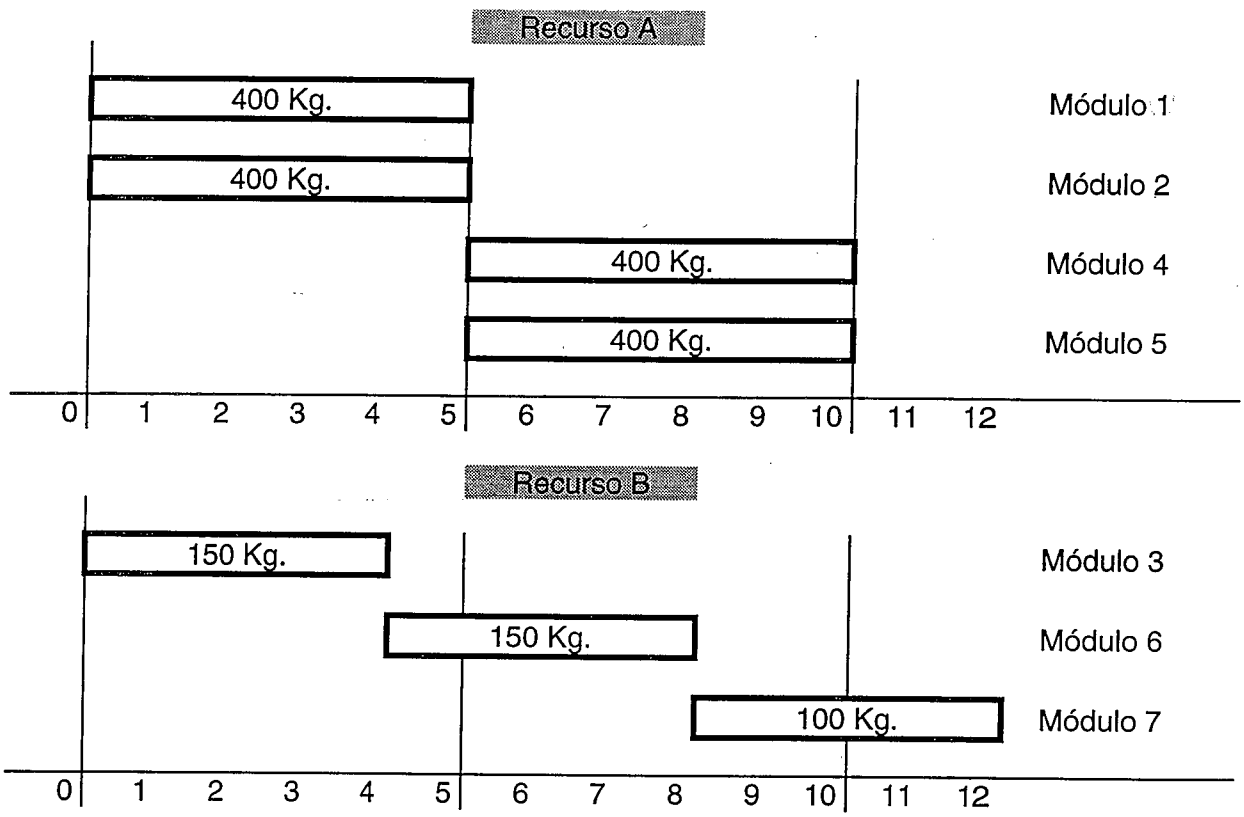
La principal cuestión en la parte del algoritmo donde tiene lugar la programación de la producción es la elección del recurso del grupo productivo al cual asignar una actividad. El nexo de unión entre actividades y recursos productivos son los conocidos como **aspectos tecnológicos**, algunos de los cuales son:

- Las capacidades totales y de cada compartimento de los recursos.
- Los límites de temperatura en algunas actividades.
- Las dimensiones de los recursos.
- Las velocidades para efectuar una actividad en un recurso.

Los aspectos tecnológicos son condiciones que permiten comprobar la total compatibilidad entre las características tecnológicas de un recurso y las requeridas por una actividad que le ha sido asignada, o bien la compatibilidad entre dos o más actividades que puedan compartir el mismo recurso durante un cierto período de tiempo.

Pongamos como ejemplo que se quiere cocer 2.000 kg de un producto, que el tiempo de cocción para este producto en condiciones normales es de 5 horas y que se dispone de dos recursos para realizar esta actividad:

- Recurso A: horno con capacidad para 800 kg del producto, separado en dos compartimentos de 400 Kg. cada uno.
- Recurso B: horno con una sola cavidad con capacidad para 150 kg de producto, aunque debido a esta característica, la duración se reduce a 4 horas.



En este caso, los 2.000 Kg. a tratar se han repartido en módulos de 400, 150 y 100 Kg. respectivamente, hasta completar la cantidad total.

3. El algoritmo de carga de máquinas

Las características a considerar en la programación de las órdenes de fabricación en los recursos de la fábrica (objetivo fundamental del algoritmo de carga de máquinas) son las siguientes:

- el proceso de asignación de una actividad o un módulo de una actividad tiene lugar dentro de una estructura de producción de capacidad limitada,
- las órdenes de fabricación se tratan globalmente, siguiendo unas prioridades específicas en una primera etapa de la programación,
- las actividades que forman parte de una orden se asignan secuencialmente a los recursos disponibles en cada caso,
- deben considerarse las incompatibilidades tecnológicas entre las características de los recursos y de las actividades,
- hay dependencias entre la fabricación de productos intermedios y finales.

Un segundo objetivo, pero no menos importante, es lograr la programación en un tiempo tan corto como sea posible.

El algoritmo de carga de máquinas tiene en cuenta los elementos presentados en la sección anterior y proporciona los resultados, la programación de la producción, impresos en forma de boletines de trabajo, que son enviados al personal de la planta para controlar la producción prevista y anotar las incidencias, que más tarde serán introducidas para una próxima ejecución del algoritmo. El algoritmo maneja los datos en memoria durante toda la programación de actividades para optimizar el tiempo de ejecución, evitando ante todo el acceso a disco, que consume una cantidad de tiempo significativa.

PRINCIPALES ETAPAS EN EL ALGORITMO:

El algoritmo de carga de máquinas original consta de cinco partes:

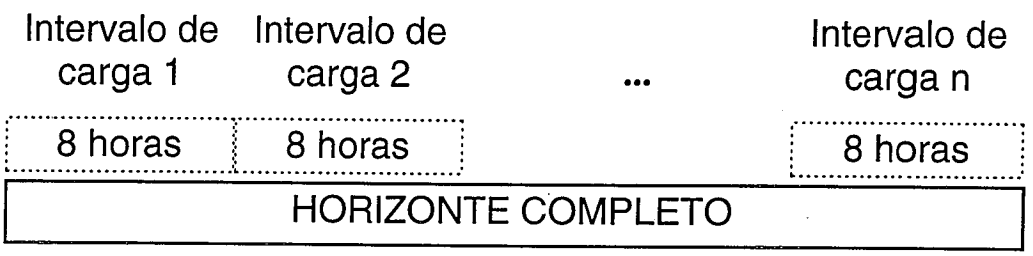
1. Preparar los datos para los cálculos de programación de las órdenes de fabricación. Obtener las precedencias entre actividades, para cada orden de producción a incluir en el programa.
2. Actualizar la estructura productiva para la iniciación de la programación: determinar qué recursos productivos están ocupados por actividades ya programadas y cuándo estarán disponibles estos recursos.
3. Determinación de la secuencia de carga de las actividades de las órdenes de fabricación: establecimiento de una secuencia de carga entre órdenes de producción y entre actividades en una misma orden de producción, que es la base sobre la cual se realiza la programación en la siguiente fase.
4. Programación de las actividades incluidas en las órdenes de producción a planificar: determinar para cada actividad a programar en qué recurso se realizará. El proceso consiste en la evaluación de cada actividad o módulo en todos los recursos productivos disponibles, y según los respectivos resultados, en la carga en el recurso productivo más apropiado.
5. Actualizar los datos de la carga de máquinas programada en soporte físico (disco).

El algoritmo propuesto para este problema es del tipo de "*dispatching*" [2], en el cual el conjunto de las operaciones programables o elegibles en un momento dado está constituido por el subconjunto de las operaciones que tienen sus precedentes ya programadas, y por lo tanto, se conocen los instantes de inicio y de finalización de éstas. En esta aplicación, no tiene porque cumplirse la propiedad de que se tomen las decisiones sobre las operaciones en un recurso en el mismo orden en que serán ejecutadas.

El algoritmo de programación incorpora los llamados intervalos de carga. Un **intervalo de carga** es una parte del horizonte de tiempo a ser programado. En alguna ocasión podría considerarse un único intervalo, en caso de considerar globalmente todo el horizonte a programar.

El intervalo de carga se usa en este método de "*dispatching*" como método de sincronismo entre las actividades de las diferentes órdenes de producción para equilibrar el tiempo total de fabricación entre ellas. Si las órdenes de producción se cargasen de principio a fin, las primeras podrían ocupar algunos recursos principales para la producción y las últimas, sufrir demoras considerables, con lo que el alimento podría deteriorarse. Esto, por supuesto, comportaría abandonar la programación en curso y tener que modificar o empezar de nuevo el cálculo de la carga de todas las actividades. Por tanto, esta actitud prudente en el diseño del algoritmo intenta evitar al máximo esta situación.

Todas las órdenes de fabricación implicadas en un programa de fabricación se van cargando progresivamente en los recursos disponibles. Así pues, todas pueden quedar un poco afectadas por tiempos de espera entre actividades. Equilibrando sus diversos retrasos, será mucho más difícil que alguna de las órdenes se demore tanto como para que no llegue a producirse dentro de los plazos de fabricación a cumplir, los cuales tienen en cuenta las fechas de deterioro de los alimentos .



Para reducir el tiempo total de cálculos se ha trabajado en dos direcciones: por un lado, en la estructuración adecuada de los datos del algoritmo necesarios para realizar los cálculos; y por otro, en una nueva definición del algoritmo anterior aprovechando las prestaciones del cálculo en paralelo.

4. El nuevo algoritmo en paralelo

De las varias partes que componen el algoritmo, solamente la cuarta incorpora el cálculo en paralelo, por ser la que comportaba mayor tiempo de cálculo en el algoritmo secuencial de partida. Esta parte consiste en la selección de una actividad de una orden de fabricación y en la evaluación numérica de su posible asignación a cada uno de los recursos productivos que la puedan realizar. Estas evaluaciones son susceptibles de ejecutarse en paralelo entre ellas porque un programa de fabricación está formado por múltiples órdenes de producción, que a la vez están integradas por numerosas actividades.

Cada una de las actividades se evalúa considerando su posible programación en todos aquellos recursos que la puedan desarrollar y se asigna una nota a cada recurso. Esta nota depende de una serie de factores, como la ocupación del recurso, la velocidad de realización de la actividad o la cantidad de producto a fabricar. Una vez evaluados todos los recursos, se escoge aquél que logra una mejor nota.

Para lanzar en paralelo estas tareas, el primer paso es seleccionar el software que permita llevarlo a cabo [5]. En este caso, para el desarrollo del algoritmo paralelo se eligió PVM (*Parallel Virtual Machine*) [1] porque permite trabajar tanto con una red de procesadores UNIX heterogéneos como con una única máquina con varios procesadores paralelos (de memoria distribuida), llamado máquina virtual. PVM es un software de libre distribución, accesible desde Internet: <http://www.epm.ornl.gov/pvm/pvm_home.html>.

Un algoritmo puede lograr resultados similares tanto en un sistema multi-CPU como en un sistema de múltiples máquinas (*cluster*). Mientras en la primera situación el tiempo de transmisión de datos es menor, el número de procesadores está limitado a la disponibilidad de la máquina; en la segunda situación, en cambio, se permite trabajar con cualquier número de procesadores. PVM también tiene estipuladas las acciones con aquellos tipos de mensajes intercambiados más a menudo [1].

Uno de los indicadores de la eficiencia de un algoritmo es el *speed-up*, que indica la mejora en tiempos de ejecución al pasar de una máquina que realiza cálculos en serie a otra que pueda realizarlos con varios procesadores [6]. El "*speed-up*" es el cociente entre el tiempo de ejecución del algoritmo en su versión secuencial y el tiempo de ejecución del algoritmo en su versión paralela, con **n** procesadores. Este valor debería aproximarse idealmente a **n**. Como conclusión, un algoritmo será tanto mejor cuanto más próximos sean los valores de *speed-up* a la cantidad de procesadores que actúan en la máquina virtual.

Otro aspecto a considerar cuando van a construirse aplicaciones en PVM es el modelo de la estructura de cálculos. Hay dos estructuras típicas [5] que se siguen en aplicaciones donde interviene el paralelismo: el modelo SPMD (*Single Process Multiple Data*), en que todos los procesos en paralelo son idénticos, aunque cada uno trabaja sobre un conjunto de datos diferentes; y el modelo *Master/Slave* (maestro y esclavos), en el cual un conjunto de procesos esclavos trabaja en una tarea para uno o varios maestros.

En el presente proyecto, se optó por el modelo *Master/Slave*, en que el conjunto de procesos esclavos llevan a cabo una tarea para un solo maestro, encargado de distribuir las diversas tareas [4]. Así pues, antes de iniciar el desarrollo del nuevo algoritmo, debían definirse las funciones del proceso maestro y las de los procesos esclavos.

Las funciones del proceso maestro básicamente son:

- Identificar y controlar los procesadores que pertenecen a la máquina virtual.
- Repartir entre los procesadores de la máquina virtual todas aquellas tareas de evaluación necesarias.

- Recibir desde los procesos esclavos los resultados de las evaluaciones.
- Escoger, de acuerdo con las evaluaciones recibidas, el que se considere como mejor recurso productivo para realizar cada actividad o módulo de actividad.

Las funciones de un proceso esclavo, por su parte, son:

- Recibir información sobre una actividad y un recurso productivo para evaluar su posible asignación.
- Hallar un período de tiempo de disponibilidad en el recurso productivo para cargar la actividad.
- Obtener el resultado de una evaluación propuesta para comparar posteriormente las diferentes opciones.
- Enviar el resultado calculado al proceso maestro.

A la hora de introducir el paralelismo en este algoritmo, surgen unos condicionantes que evitan poder llevar a cabo una realización completa de cálculos en paralelo en cada momento de la ejecución. Estos aspectos conducen a plantear una estructura de los cálculos basada en los llamados **niveles de paralelismo**, que se asociará a las siguientes circunstancias:

1. Primer nivel de paralelismo: paralelismo en las evaluaciones de actividades en cada uno de los recursos. Las actividades en una cierta orden de fabricación deben cargarse de manera secuenciada. Por lo tanto, los cálculos referidos a una actividad o módulo no pueden comenzar si antes no ha sido cargado el módulo o actividad previa.

2. Segundo nivel de paralelismo : paralelismo en las evaluaciones de actividades y de órdenes en cada uno de los recursos. Las actividades de órdenes de fabricación diferentes pueden ocasionar conflictos en caso que los cálculos se refieran al mismo recurso productivo. Esta posibilidad comporta que cuando una actividad va a asignarse a un recurso productivo, éste debe tener su información actualizada.

PRIMER NIVEL DE PARALELISMO

Si sólo se considera el primer nivel de paralelización, las órdenes de fabricación a programar se cargan en la secuencia previamente fijada en una etapa anterior del algoritmo. Por otro lado, y debido a las relaciones de precedencia entre actividades, las actividades de cada orden se cargan también siguiendo una cierta ordenación. Así pues, en el curso de la ejecución del algoritmo los procesos esclavos evalúan, de manera paralela, los recursos productivos candidatos a ser ocupados por una actividad de la orden de fabricación en curso de asignación.

El proceso maestro, pues, debe esperar todos los resultados de las evaluaciones de una actividad sobre cada posible recurso productivo. Una vez recogidas estas evaluaciones, se selecciona el recurso más apropiado, la actividad se carga y los recursos productivos involucrados en evaluaciones de la siguiente actividad pueden enviarse a los procesos esclavos.

Analizando los valores del *speed-up*, se observó que eran relativamente bajos: mientras algunas actividades podían asignarse a diversos recursos, para lo cual la introducción del cálculo en paralelo era positivo, en caso que este número de recursos fuera escaso (dos o tres) el algoritmo perdía eficiencia. Por lo tanto, debía complementarse el grado de paralelismo existente en los cálculos, y como consecuencia, incrementarse el *speed-up*.

SEGUNDO NIVEL DE PARALELISMO

Las actividades de cada orden de fabricación se cargan siguiendo una ordenación prefijada. Al igual que antes, las órdenes de fabricación se ordenan, pero no se obliga a terminar la carga de una orden de fabricación antes de comenzar a cargar actividades de otra. Así pues, en el curso de ejecución del algoritmo los procesos esclavos pueden evaluar, en paralelo, los recursos productivos candidatos a ser ocupados por actividades de varias órdenes.

El proceso maestro no debe esperar todos los resultados de la evaluación de una actividad sobre los posibles recursos antes de enviar a los procesos esclavos evaluaciones de los recursos involucrados en otra actividad, a menos que esta última actividad tenga que programarse ya en el siguiente intervalo de carga.

Este nuevo modelo de evaluación de órdenes en paralelo aumentaba la eficiencia porque las diversas actividades implicaban mayores posibilidades de cálculo en las órdenes de fabricación programadas simultáneamente. Y esta eficiencia podía incrementarse incluso más al estar programando actividades en varios puntos de la estructura de producción, y evitar posibles repeticiones en los cálculos, como explicamos seguidamente.

CONFLICTOS ENTRE ÓRDENES DE PRODUCCIÓN

En la aplicación del algoritmo, no obstante, pueden surgir conflictos entre órdenes de producción en paralelo cuando algunas de sus actividades compitan por el mismo recurso productivo. Cuando una actividad va a asignarse a un recurso, debe comprobarse que ninguna otra actividad se ha cargado desde que tuvo lugar su evaluación hasta que es definitivamente cargada en el recurso elegido. Si el contexto que afecta al recurso ha cambiado, la evaluación de carga podría no ser correcta.

Ante este problema, se pueden tomar diversas alternativas [3], entre las cuales se consideraron dos posibles soluciones: la primera, basada en un sistema de bloqueos de los recursos; la segunda, basada en un sistema de recálculo de las evaluaciones. Finalmente, se optó por esta última opción porque utilizando los bloqueos de los recursos, la relación de los tiempos de cálculo *speed-up* no mejoraba substancialmente y además se corría el riesgo de alcanzar una situación de bloqueo general del sistema, hecho posible al ejecutar acciones paralelas dependientes entre sí [3].

SISTEMA DE RECÁLCULOS

El sistema de recálculos, causado por la existencia de los dos niveles de paralelismo, considera la evaluación de varias actividades diferentes pertenecientes a más de una orden de fabricación, sobre un mismo recurso productivo a la vez. Cualquier evaluación puede enviarse a calcular en algún momento, aunque siempre con la posibilidad de tener que recalcularse más tarde. El recálculo tendrá lugar cuando un recurso sea el más adecuado para llevar a cabo dos actividades diferentes que se están programando.

Analizados los resultados, se observó como el *speed-up* mostraba una gran mejora respecto al único nivel de paralelismo, aunque la presencia de intervalos de carga quizá no permitía el máximo grado de paralelismo. El número de recálculos no era excesivo y, además, era inferior normalmente al número de bloqueos.

5. **Resultados**

El algoritmo secuencial original ha tenido que adaptarse para ser paralelamente eficiente, es decir, que la cantidad de cálculos simultáneos sea compensada con una reducción significativa en el tiempo de ejecución. Ello obligó a permitir evaluar simultáneamente asignaciones de actividades correspondientes a diversas órdenes de fabricación, aunque esto pueda comportar que se evalúe sobre un mismo recurso más de una actividad. Esta posibilidad comporta contratiempos, los recálculos en las evaluaciones. La cantidad de recálculos depende del número y del tipo de procesadores en que se ejecute el algoritmo, aunque si se analizan los tiempos, esta incidencia apenas perjudica la duración total de la ejecución.

El código del nuevo algoritmo se ha escrito en lenguaje de programación C, y para el cálculo en paralelo se utilizan aquellas rutinas de paralelización propias del software PVM (*Parallel Virtual Machine*).

DURACIONES DE LA EJECUCIÓN DEL NUEVO ALGORITMO

El objetivo de reducir la duración de la ejecución del Algoritmo de Carga de Máquinas de varias horas a una duración de entre 1 y 2 horas, para que el algoritmo pueda ser nuevamente ejecutado en caso de incidencias en la planta, se ha superado con creces. Una vez introducido el cálculo en paralelo para realizar la programación de la producción, el tiempo alcanzado en la ejecución del nuevo algoritmo ha logrado situarse en minutos, debido a varios factores.

Para empezar, se depuró la gestión de datos optimizando el tiempo en operaciones no directamente vinculadas a la programación de la producción, y se emprendió la estructuración de la memoria, a fin de disponer de aquella información estrictamente necesaria para el funcionamiento en paralelo. Como último paso, se podía atacar con la mejora fundamental en el algoritmo: la utilización del cálculo paralelo para incrementar la velocidad de obtención de la programación esperada. Esta tríada de elementos adecuadamente equilibrados ha permitido lograr los resultados de "*speed-up*" que seguidamente se detallan.

Los resultados proporcionados son las duraciones obtenidas ejecutando el nuevo algoritmo sobre los datos referidos a cinco programaciones semanales. La plataforma sobre la cual se realizó esta experiencia computacional es una Silicon Graphics (con un procesador MIPS de 64 bits) utilizando hasta 8 procesadores actuando conjuntamente.

El proceso maestro y el primer proceso esclavo comparten procesador ya que los períodos de actividad de ambos procesos apenas se solapan. En cada una de las cinco pruebas realizadas, se evaluó la duración del algoritmo utilizando 1, 2, 4 y 8 procesadores, respectivamente. Los resultados facilitados se obtienen como media de estas pruebas.

El proceso maestro consta básicamente de dos partes: en la primera, se capta la información necesaria para la programación de la producción de los datos almacenados en un fichero previamente generado y las tareas se preparan para ser tratadas (en estas acciones, la paralelización no interviene); en la otra parte, se gestiona el algoritmo en paralelo usando el modelo de maestro y esclavos. Los tiempos correspondientes a la primera parte no se consideran a efectos de "*speed-up*" porque no afectan a la ejecución en paralelo. Sí, en cambio, deben ser considerados a la hora de comparar las duraciones totales de la nueva versión del algoritmo con las duraciones de la anterior versión.

Nº procesadores	1	2	4	8
Parte inicial	3560	3955	4790	5429
Parte paralelismo	92808	63881	49828	34795
Tiempo total	96368	67836	54618	40224

Los tiempos de la tabla anterior son bastante representativos del tiempo de actividad de cada procesador, puesto que se ha incorporado una sentencia interna que recoge los tiempos de puesta en marcha y parada de cada procesador. Seguidamente, se desglosa la duración de actividad del proceso maestro y de los procesos esclavos para cada uno de los cuatro casos:

Nº procesadores	1	2	4	8
Tiempo total	96368	67836	54618	40224
Tiempo actividad maestro	1021	1091	1408	2196
Tiempo medio actividad esclavos	87292	59154,5	38831,75	17786,38
Tiempo actividad esclavo 1	87292	61888	47928	33365
Tiempo actividad esclavo 2		56421	40347	26778
Tiempo actividad esclavo 3			34867	21160
Tiempo actividad esclavo 4			32185	16498
Tiempo actividad esclavo 5				12904
Tiempo actividad esclavo 6				11346
Tiempo actividad esclavo 7				10378
Tiempo actividad esclavo 8				9862

La utilización media de los procesos (el % como resultado del cociente entre el tiempo de utilización de los procesos respecto al tiempo total de ejecución) en el proceso maestro se sitúa entre el 1% y el 6%, mientras que en los esclavos, ronda el 90%. Estos datos pueden indicar, por un lado, que es sensato que el proceso maestro y un proceso esclavo compartan procesador, y por otro, que la estructura del algoritmo paralelo es adecuada para el uso eficiente de los procesadores.

La tendencia del "*speed-up*" cuando el número de procesadores (procesos esclavos) aumenta, con la actual cantidad de órdenes y los actuales factores para la evaluación de actividades en recursos, se aleja pronto del "*speed-up*" ideal. A la vista de estos resultados, la aplicación difícilmente podría ser completamente eficiente desde el punto de vista del paralelismo, es decir, de la utilización casi al 100% de todos los procesadores implicados en los cálculos. Observando la siguiente tabla, se puede concluir que aumentando el número de procesadores, se incrementa el "*speed-up*", si bien con una tendencia a reducir la tasa de incremento:

Nº procesadores	1	2	4	8
Tiempo del proceso maestro (ms)	92808	63881	49828	34795
Speed-up	1,000	1,453	1,863	2,667
Utilización media del procesador	1	0,73	0,47	0,33

No obstante, en la empresa final usuaria del algoritmo no se dispone de tecnología suficientemente potente como la utilizada para obtener los anteriores resultados, sino que cuenta con un servidor con múltiples procesadores de menores prestaciones. Esto implica que las duraciones de ejecución se incrementen sensiblemente. Este servidor, que dispone de la arquitectura y el software necesario para ejecutar esta aplicación, se encuentra conectado en red a unos cuantos PCs también equipados convenientemente.

VALIDACIÓN DE LA PROGRAMACIÓN OBTENIDA

A fin de determinar un índice de calidad cuantitativa para los resultados del algoritmo, se considerará, por un lado, la programación de actividades resultante de aplicar el nuevo algoritmo y, por otro, aquellas estadísticas surgidas de la ejecución del algoritmo.

PROGRAMACIÓN DE LOS RECURSOS PRODUCTIVOS

Los **recursos productivos**, definidos según las necesidades del usuario final, siguen unos procesos muy heterogéneos. Esto implica una gran variedad de aspectos, como la importancia de considerar los turnos de trabajo u otras propiedades similares, que en etapas más avanzadas del uso del algoritmo podrían reportar mejores resultados del paralelismo. Estudiando la ocupación media de los recursos, fijado un número de turnos y de paradas por descanso del personal, se puede optimizar no sólo el uso de máquinas, sino también del personal destinado a estos recursos.

Al analizar la carga de trabajo en diferentes recursos de producción, la asignación de actividades estáticas a los recursos modulares tiene perfectamente en cuenta los criterios de capacidad disponible total del recurso, la capacidad por módulo (en el supuesto de recursos modulares de producción) y el número de módulos del recurso.

Según el número de órdenes de producción, la cantidad de actividades por orden y el tipo de proceso utilizado, la duración de la carga de máquinas será más corta o más larga. Sin embargo, el tiempo usado por los procesos esclavos para efectuar una evaluación se ha reducido suficientemente para permitir ejecutar el algoritmo muy frecuentemente si la fábrica requiere cambios en la programación de las órdenes de fabricación.

El grado de paralelismo en el algoritmo también queda patente en la programación de órdenes de producción con la carga en curso, aquéllas con algunas actividades que deben programarse de nuevo después de un cierto tiempo de trabajo. Normalmente se trata de actividades de órdenes regeneradas a partir de antiguos archivos de programaciones. El nuevo fichero con las órdenes a programar actualiza las asignaciones de actividades a recursos en el instante inicial de la nueva carga de máquinas (algunas actividades programadas en la anterior ejecución se están realizando en el nuevo instante de inicio) y deja pendientes de programar el resto de actividades de cada orden.

Los resultados de la programación obtenidos en una segunda ejecución del algoritmo respecto a los alcanzados en la anterior ocasión no han de ser obligatoriamente idénticos, porque las evaluaciones y las asignaciones dependen de la distribución de cálculos entre los procesadores paralelos, que puede variar en cada puesta en marcha del algoritmo.

Si se tienen en cuenta los intervalos de carga, necesarios para equilibrar el tiempo de fabricación total entre diferentes órdenes de producción, el número de intervalos suele ser bastante alto (si se toman los valores habituales en la fábrica, de 6 horas por intervalo), debido a unas pocas órdenes que sobrepasan habitualmente los 15 días. Es decir, hay unas órdenes que tienen una duración total mucho mayor que el resto. Estas programaciones tan prolongadas están ocasionadas por actividades realmente largas (de unos 30 días, período de tiempo en que mantienen ocupados los secaderos) en comparación con las demás. Esta circunstancia se manifiesta como adversa al objetivo de máximo grado de paralelismo.

ESTADÍSTICAS DEL ALGORITMO PARALELO

Una vez finaliza la ejecución del algoritmo pueden extraerse algunos datos estadísticos y analizarse: el número mínimo de cálculos a repartir entre los procesos esclavos, la cantidad de estos cálculos que se han debido repetir, etc.

Un dato estadístico útil es el número de actividades por carga de máquinas. Hay alrededor de unas 800 actividades subdivididas en unos 5.000 módulos o entregas, que es variable según el número de órdenes de fabricación que empiezan cada semana en la planta. Como el número medio de recursos por evaluación en cada entrega se sitúa entre 3 y 4, el número de cálculos a distribuir estará entre los 14.000 y los 20.000, valor que se convierte en el **número mínimo de cálculos** a realizar.

Por otro lado, hay una cantidad de **cálculos repetidos** porque la evaluación obtenida en el intento de asignación de una actividad a la carga de un recurso no es representativa en el momento de la asignación definitiva. En este caso, la incertidumbre en el reparto de tareas entre los esclavos no permite un control exhaustivo sobre este factor. No obstante, este número de repeticiones de tareas no ha logrado reducirse a pesar de algunos intentos encaminados a minimizar esta cantidad de asignaciones.

Otro parámetro estadístico en la carga de máquinas es el número de **situaciones de espera** que suceden a lo largo de la ejecución, cuando algún proceso esclavo está libre y ningún cálculo figura como pendiente de evaluación. Esta cantidad aumenta más que linealmente al disponer de un número cada vez mayor de procesos esclavos, porque cuando se calcula una única orden al final de un intervalo de carga sólo se permite usar un único esclavo. Sin embargo, esta cantidad de situaciones de espera se mantiene muy similar en caso que la duración del intervalo de carga se amplíe.

Como el usuario final no usa fechas de entrega como referencia para los lanzamientos de órdenes en su estructura productiva, para evaluar la calidad del algoritmo de Carga de Máquinas será necesario referirse a la duración mínima de una orden de fabricación, que considere las actividades correctamente secuenciadas. Es decir, se utilizará un porcentaje de desviación entre la "duración ideal" (sin considerar superposiciones entre actividades ni actividades de otras órdenes) y la "duración verdadera" (lograda con el algoritmo de carga de máquinas).

Los resultados obtenidos con la nueva aplicación del algoritmo sobre un conjunto de datos tienen una desviación en promedio del 33%. Si estos valores relativos los analizamos de manera absoluta, se puede elaborar la siguiente tabla, en que clasificamos las órdenes según su duración mínima:

Duración mínima de una orden	Duración programada de una orden
entre 0 y 24 horas	entre 0 y 36 horas
entre 24 y 48 horas	entre 32 y 88 horas
entre 48 y 72 horas	entre 52 y 101 horas
entre 72 y 90 horas	entre 73 y 174 horas
entre 90 y 120 horas	entre 98 y 180 horas

Si se observa la programación elaborada para cada recurso productivo, se constata la existencia de dos grandes bloques de recursos: por un lado, aquellos recursos estratégicos, comunes a todas las líneas productivas, que actúan como elementos que marcan la mayor o menor duración de una orden; y por otro lado, otros recursos más específicos de cada línea de productos.

6. CONCLUSIONES

La revisión del anterior algoritmo de carga de máquinas ha permitido introducir la paralelización en los cálculos para reducir los tiempos globales de ejecución. No obstante, la detección de algunas otras deficiencias, que también se han resuelto, ha complementado los logros obtenidos y ha permitido conseguir duraciones de ejecución competitivas.

Una primera reducción notable de tiempo fue lograda con la utilización adecuada de la memoria con los datos necesarios para llevar a cabo la programación de la producción. Superada esta etapa, se preparó la estructura del algoritmo siguiendo el modelo de proceso maestro y procesos esclavos, pero se lograron bajos rendimientos en paralelismo si se utilizaba un solo nivel. A lo cual, se respondió añadiendo un segundo nivel, que permitía acercarse más a los "*speed-up*" usuales en aplicaciones que demuestren positivamente la validez del paralelismo como respuesta a sus problemas.

Los resultados logrados en tiempo de ejecución, incluso en ordenadores con procesadores estándares, son calificables como más que satisfactorios y plantean la posibilidad de incorporar progresivamente las suficientes prestaciones complementarias para llegar a poner en práctica la aplicación de este algoritmo en tiempo real.

No obstante, la aplicación de algoritmos que introducen elementos del tipo "*dispatching*" y que procuran avanzar sincronizadamente en diferentes frentes (en este caso, en la programación de diversas órdenes) ha demostrado la difícil capacidad de superación de ciertos límites al aplicar la paralelización en los procesos de cálculo. En esta aplicación, la presencia de los intervalos de carga o de los recálculos necesarios, si se realizan evaluaciones en paralelo sobre un mismo recurso, evitan la posibilidad de alcanzar mejores resultados en el paralelismo debido a las dependencias entre las diversas evaluaciones [6].

Bibliografía:

[1] Breshears, Clay. (1995). "A beginner's guide to PVM- Parallel Virtual Machine". URL, <http://csep1.phy.ornl.gov/CSEP/PVM/PVM.html>. University of Tennessee.

[2] Companys, Ramón; Corominas, Albert. (1996). Organización de la producción II. Dirección de operaciones 4. Edicions UPC.

[3] Feitelson, Dror; Rudolph, Larry. (1995). Job Scheduling Strategies for Parallel Processing. Springer.

[4] Jájá, Joseph. (1992). An Introduction to Parallel Algorithms. Addison-Wesley Publishing Company.

[5] Labarta, Jesús; Valero, Mateo; Gregoris, Luis. (1995) "Introduction to parallel computing & PVM", CEPBA Working Paper.

[6] Zenios, Stavros A. (1994). "Parallel and Computing in the Practice of Management Science". Interfaces 24: 5 set-oct, pp. 122-140.