Soporte en la planificación para la concurrencia y el paralelismo en el Sistema Operativo Mach

Scheduling Support for Concurrency
and Parallelism in the Mach Operating System

Autores: Aitor Acedo y Sergio Paracuellos

Indice

- Conceptos iniciales básicos
- Introducción al problema
- Trabajos relacionados
- Desarrollo del contenido
  - Resumen general
  - Modelos de programación
  - Soporte de la concurrencia en el planificador
  - Asignación de procesadores
    - Diseño
    - Implementación
- Conclusiones
- Notas

Conceptos iniciales básicos I

Concurrencia: existencia simultánea de varios procesos en ejecución. No implica que estén ejecutándose simultáneamente.

Paralelismo: caso particular de la concurrencia, ocurre cuando se produce la ejecución simultánea de varias instrucciones.

Planificador de sistema (scheduler): parte del núcleo que se encarga de la asignación de recursos para la ejecución de procesos "equitativamente". (en kernel monolítico)

Mach: sistema operativo microkernel desarrollado en los 80 en la Universidad de Carnegie. (nuevas tendencias)

Microkernel: estrategia de diseño del núcleo en la que se reduce a la mínima expresión el kernel, dejando fuera funciones no críticas.

Conceptos iniciales básicos II

Planificación Gang: Es el concepto de planificación como un conjunto de procesos sobre un conjunto de procesadores.

Planificación a corto plazo: Determina cuál es el próximo proceso a ejecutar. Es invocada cada vez que ocurre un evento que pueda causar una suspensión.

Planificación a largo plazo: Determina qué nuevos programas son aceptados para ser procesados por el sistema, es decir, determina el grado de multiprogramación.

Paralelismo de grano fino: paralelismo inherente en un único flujo de instrucciones. El intervalo de sincronización es de <20 instr's.

Introducción al problema I

Multiprocesador => muchas aplicaciones a la vez (orígenes)
Multiprocesador #> 1 aplicación + rápido
Programación paralela => 1 aplicación + rápido

Los planificadores tradicionales de sistemas operativos de tiempo compartido no son adecuados para programas concurrentes y paralelos, los cuales, requieren nuevas técnicas como por ejemplo: la *asignación de procesadores*, y la *no intervención de la planificación*.

Introducción al problema II

Un claro ejemplo de programas que no encajan en el concepto tradicional de planificación serían, por ejemplo, los programas paralelos que necesitan un número fijo de procesadores para ejecutarse.

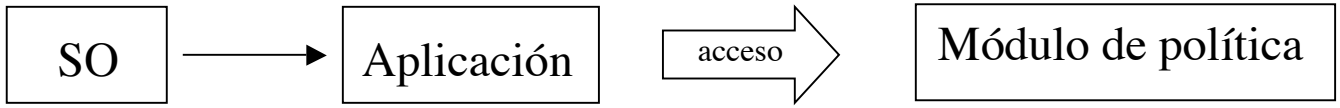
El planificador tendría que asignarle ese número de procesadores al programa en cuestión, porque de lo contrario el rendimiento del mismo bajaría considerablemente. Este y otros problemas hacen necesario que tengamos que inventarnos nuevas técnicas para la planificación de las tareas en máquinas que soportan paralelismo y concurrencia.

Trabajos relacionados I

Separación del mecanismo de política: (*ver Notas)

(Planificación interna de la aplicación)

Problemas:



Otro problema fue que la mayoría de las aplicaciones no utilizaban la flexibilidad disponible, debido a la complejidad de los módulos de política.

Trabajos relacionados II

La separación adoptada en Mach, del mecanismo de planificación y de la política evita los dos problemas citados anteriormente.

Las decisiones de asignación de procesadores son tan infrecuentes que amortizan los costes de la comunicación.

La implementación de la política reside en un servidor que está implementado una vez en cada sistema, en lugar de en un módulo que podría ser configurado para cada aplicación.

Desarrollo del contenido I

Resumen general:
Este artículo describe la incorporación de la asignación y control de procesadores dentro del sistema operativo Mach.
El enfoque de diseño divide la implementación en 3 componentes:

1) Mecanismos básicos de implementados en el kernel. (mecanismo)
2) Política a largo plazo implementada en un servidor. (política)
3) Implementación opcional de usuario de la política a corto plazo.

El aislamiento de la política a largo plazo proporciona la ventaja de la separación del mecanismo de política, disminuyendo las desventajas encontradas en implementaciones anteriores de este principio en multiprocesadores.

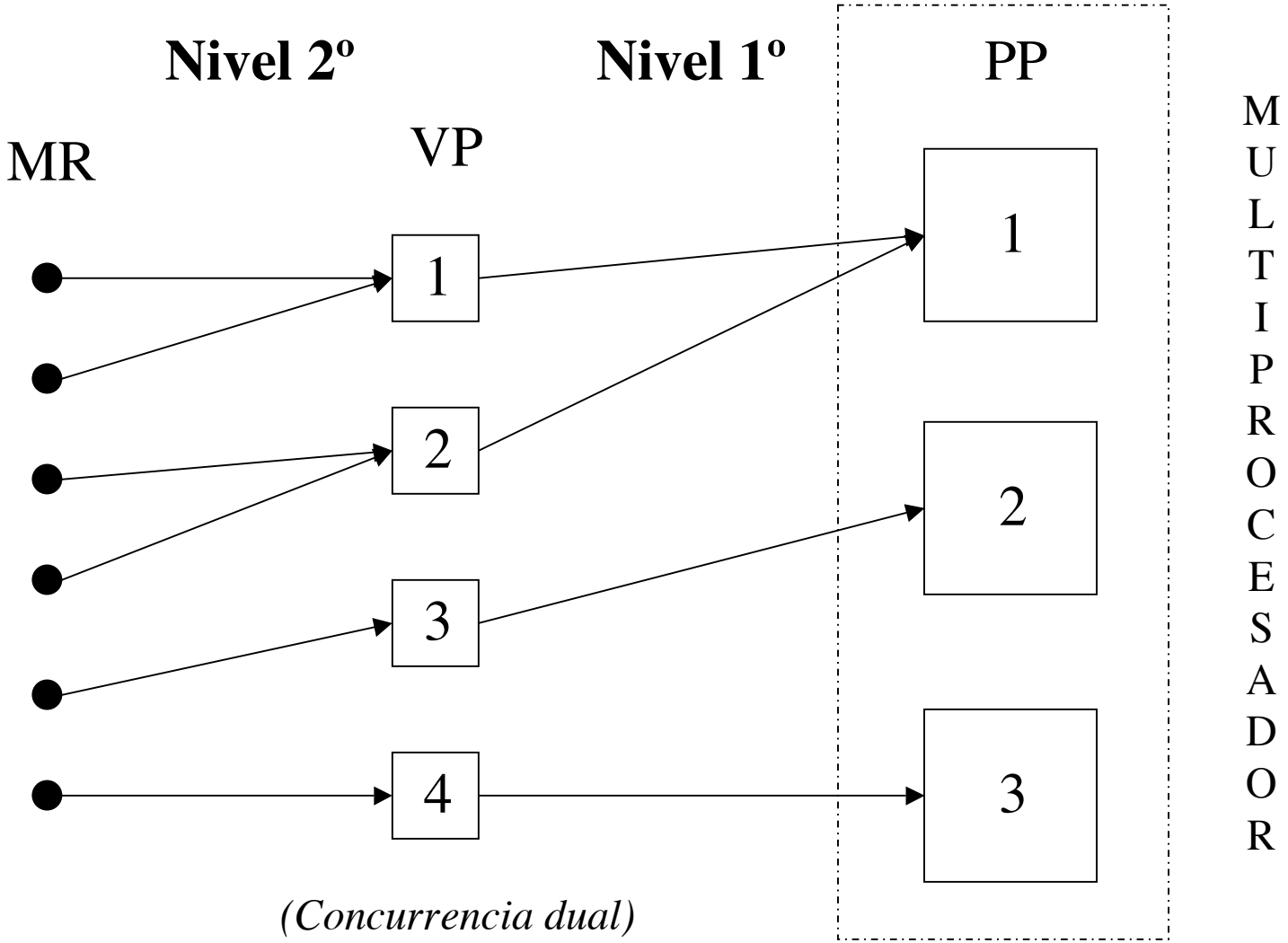
Desarrollo del contenido II

Modelos de programación:
Los modelos de programación de aplicaciones pueden introducir concurrencia más allá del paralelismo del hardware, en dos niveles.

Un sistema operativo puede introducir concurrencia multiplexando entidades planificables por separado, llamadas *procesadores virtuales*, en procesadores hardware.

El segundo nivel de concurrencia puede ser introducido por las librerías a nivel de usuario o entidades a nivel de lenguaje, multiplexándolas en procesadores virtuales (VP). Esas entidades las llamaremos multirutinas (MR).
Denotaremos a los procesadores físicos como (PP).

Desarrollo del contenido III



(Concurrencia dual)

Desarrollo del contenido IV

Los modelos de programación concurrente y paralela pueden ser clasificados por la relación existente entre las MR's, los VP's y los PP's.
Esta tabla indica el tipo de relaciones posibles:

Relación	Modelo
MR = VP = PP	Paralelismo puro
MR > VP = PP	Concurrencia de usuario
MR = VP > PP	Concurrencia de sistema
MR > VP > PP	Concurrencia dual

Desarrollo del contenido V

Soporte de la concurrencia en la planificación:
Las aplicaciones cuando usan más procesadores virtuales que físicos pueden beneficiarse de la entrada de usuario para las decisiones de planificación.
Los usuarios tendrían información sobre que VP's deberían estar ejecutándose y cuales no.
Los tipos de consejos que acepta el planificador son de dos tipos:
• *Consejos de disuasión*: pueden ser moderados, fuertes y absolutos, indican que el actual hilo no debería estar en ejecución.
• *Consejos de no intervención*: indican que un determinado hilo debería ejecutarse en vez del que se encuentra ejecutándose.

Desarrollo del contenido VI

Los consejos de disuasión son útiles para la optimización de la sincronización en memoria compartida de aplicaciones empleadas en sistemas de concurrencia.

La segunda clase de consejos, de no intervención de planificación, envían directamente al procesador el hilo especificado, evitando los mecanismos internos de planificación.

Experimentos realizados demuestran que aplicaciones de sistemas concurrentes requieren soporte de fuerte disuasión y que la no intervención de la planificación es efectiva si la información requerida está disponible.

Desarrollo del contenido VII

Asignación de procesadores:
La planificación Gang, garantiza la planificación simultánea de componentes de una misma aplicación, se usa para la asignación de procesadores en sistemas operativos multiprocesador.
Es necesaria para aplicaciones paralelas de grano fino cuyo rendimiento se degrada severamente cuando cualquier parte de la aplicación no se está ejecutando.

Diseño:
El factor dominante en el diseño de la asignación de procesadores es la *flexibilidad* debido al soporte que ofrece Mach para multitud de aplicaciones, lenguajes, modelos de programación y arquitecturas.

Desarrollo del contenido VIII

Esta flexibilidad tiene varios aspectos:

• No solamente tenemos que permitir la asociación de un hilo a un procesador sino que deberemos permitir asociar un conjunto de hilos a un conjunto de procesadores para las aplicaciones que utilicen sistema de concurrencia.

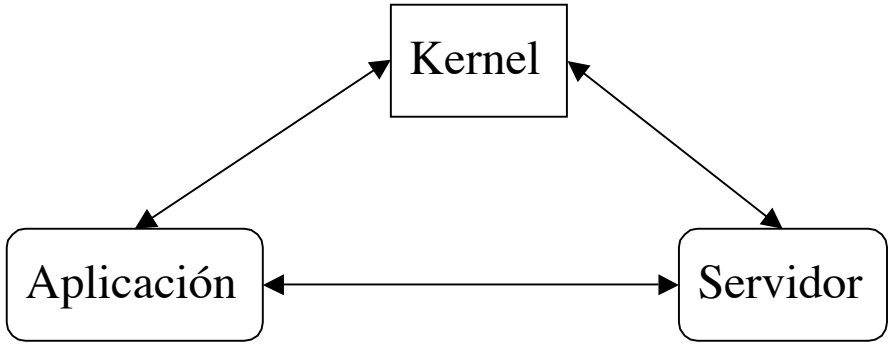
• Posibilitaría el cambio de políticas de asignación sin que sea necesaria la recompilación del kernel.

• Se ofrecería un completo control de que hilos se ejecutan sobre que procesadores teniendo una asignación por defecto.

La asignación añade dos nuevos objetos al interface del kernel Mach, el ‘procesador’ y el ‘conjunto de procesadores’.

Desarrollo del contenido IX

Componentes de la asignación de procesadores:



La responsabilidad para la asignación y uso de los procesadores dedicados está dividida entre la aplicación, el servidor y el kernel. La aplicación controla la asignación de tareas e hilos a conjuntos de procesadores. El servidor controla la asignación de procesadores a conjuntos de procesadores. Y el kernel hace lo que el servidor y la aplicación le han dicho que haga.

Desarrollo del contenido X

Ahora vamos a mostrar un ejemplo de cómo una aplicación asignaría 6 procesadores:

1. **Aplicación => kernel** Crear conjunto de procesadores
2. **Aplicación => Servidor** Petición de 6 procesadores para el conjunto
3. **Aplicación => kernel** Asignación de hilos al conjunto
4. **Servidor => kernel** Asignación de los procesadores al conjunto
5. **Aplicación** usa los procesadores
6. **Aplicación => Servidor** Finalizado con los procesadores (opt)
7. **Servidor => kernel** Reasignación de procesadores

Desarrollo del contenido XI

Implementación:
La implementación en el kernel del conjunto de procesadores extiende el planificador de tiempo compartido de Mach.

La estructura de datos para cada conjunto de procesadores contiene una cola de ejecución que será global para los procesadores del conjunto.

También contendrá información requerida para el algoritmo de planificación, una lista con las estructuras de datos de las tareas asignadas y un cerrojo de exclusión mutua para controlar el acceso a la estructura de datos.

Además mantendrá una lista de los procesadores libres dentro del conjunto para poder asignar tareas independientes del conjunto.

Conclusiones

Las conclusiones del autor:

Muchas aplicaciones concurrentes y paralelas pueden ser planificadas aceptablemente (con menor rendimiento) por los métodos de tiempo compartido tradicionales.

El uso de procesadores dedicados es para conseguir un rendimiento aceptable de determinadas aplicaciones paralelas.

Para las aplicaciones concurrentes, el rendimiento de sincronización y comunicación puede ser mejorado recibiendo información de planificación especifica de la aplicación. (consejos)

Conclusiones

Nuestras conclusiones:

Como el propio autor indica nos hubiera parecido un artículo más completo si la solución aportada por el artículo resolviera por completo la problemática para arquitecturas de acceso no uniforme a memoria, (con tiempos de acceso dependientes de la distancia al procesador), ya que no se soluciona completamente el problema del balanceo de carga para estas máquinas.

En textos en un idioma diferente al nuestro siempre aparecen términos que resultan difíciles de traducir, en nuestro caso el más complicado sin duda ha sido ‘handoff scheduling’ que hemos traducido por ‘no intervención de la planificación’.

Notas

En el punto de ‘Trabajos Relacionados‘ se hace referencia a las siguientes obras:

W. A. Wulf, R. Levin, and S.P. Harbison, Hydra/C.mmp: An Experimental Computer System, McGraw-Hill, New York, 1981

E. F. Gehringer, D. P.Siewiorek, and Z. Segall, Parallel Processing: The Cm* Experience, Digital Press, Maynard, Mass., 1987