# 10.1

La programación en sistemas con múltiples procesadores o núcleos presenta desafíos específicos. Hay diferentes tipos de configuraciones de sistemas multiprocesador, como aquellos donde cada procesador tiene su propia memoria y otros recursos, o sistemas donde comparten una memoria común bajo el control de un solo sistema operativo.

Cuando se trata de programar en estos sistemas, es importante considerar la "granularidad", que se refiere a la frecuencia con la que los procesos deben sincronizarse entre sí. Hay varios niveles de granularidad:

* **Paralelismo independiente**: Los procesos no necesitan sincronizarse y pueden ejecutarse de manera independiente, como en sistemas donde múltiples usuarios realizan tareas separadas.
* **Paralelismo de grano grueso y muy grueso**: Los procesos se sincronizan en niveles más altos, lo que permite que varios procesos se ejecuten concurrentemente con poca necesidad de comunicación constante.
* **Paralelismo de grano medio**: Requiere una coordinación más estrecha entre los subprocesos de una aplicación, con una sincronización más frecuente debido a interacciones más complejas.
* **Paralelismo de grano fino**: Implica una sincronización detallada entre procesos, necesario para aplicaciones muy especializadas y altamente paralelas.

La programación eficiente en sistemas multiprocesador implica decisiones sobre cómo asignar procesos a los procesadores disponibles, cómo gestionar la multiprogramación y cómo enviar procesos de manera efectiva. Estas decisiones dependen de la naturaleza de las aplicaciones y de la cantidad de recursos disponibles.

En un contexto de multiprogramación en un procesador individual, surge la pregunta de si es eficiente asignar un procesador a un solo proceso estáticamente durante su vida útil. Esto puede resultar ineficiente, especialmente cuando los procesos se bloquean frecuentemente por E/S o por sincronización. En un multiprocesador, donde hay varios procesadores disponibles, la necesidad de alta utilización individual de cada procesador disminuye, priorizando en cambio el rendimiento global de las aplicaciones. La multiprogramación se convierte entonces en una herramienta para equilibrar la carga y maximizar el rendimiento promedio.

El despacho de procesos en multiprocesadores se simplifica en comparación con sistemas monoprocesador más complejos, utilizando estrategias como FCFS (primero en llegar, primero en ser servido) o round robin, que pueden ser adecuadas dependiendo del número de procesadores y la variabilidad en los tiempos de servicio de los procesos.

La programación de subprocesos se discute como una forma eficiente de manejar aplicaciones con paralelismo a nivel de subprocesos, especialmente en entornos multiprocesador donde diferentes subprocesos pueden ejecutarse simultáneamente en varios procesadores para mejorar el rendimiento global. Se mencionan estrategias como la carga compartida, la programación grupal, la asignación de procesadores dedicados y la programación dinámica como enfoques para gestionar subprocesos eficientemente.

La carga compartida, por ejemplo, distribuye uniformemente la carga entre los procesadores disponibles, asegurando que todos estén ocupados siempre que haya trabajo por hacer. Sin embargo, esto puede llevar a cuellos de botella si no se maneja adecuadamente la exclusión mutua en la cola global de subprocesos listos.

Finalmente, se explora la programación de pandillas, que consiste en programar conjuntamente subprocesos relacionados de una misma aplicación en diferentes procesadores, minimizando así los cambios de contexto y mejorando la eficiencia de la sincronización entre subprocesos.

La asignación de procesadores dedicados implica asignar específicamente a cada subproceso de una aplicación un procesador que permanece exclusivamente dedicado a ese subproceso hasta que la aplicación finaliza. Aunque esto podría parecer ineficiente, especialmente si un subproceso está bloqueado esperando entrada o sincronización, tiene beneficios potenciales. En sistemas con muchos procesadores, la utilización individual del procesador no es tan crítica, y evitar los cambios de contexto puede acelerar significativamente la ejecución de ciertas aplicaciones, como se demostró en estudios comparativos de multiplicación de matrices y transformada rápida de Fourier.

En contraste, la programación dinámica permite ajustar dinámicamente el número de subprocesos activos en función de la carga y los recursos disponibles. Este enfoque involucra tanto al sistema operativo como a la aplicación en la asignación de procesadores, permitiendo una mayor flexibilidad y potencialmente mejorando la utilización de recursos en sistemas multiprocesador o multinúcleo.

Además, la programación de subprocesos en sistemas multinúcleo debe considerar cómo asignar subprocesos para maximizar el uso eficiente de la memoria caché compartida entre núcleos adyacentes. Esto es crucial para minimizar los accesos a la memoria fuera del chip, lo cual es costoso en términos de rendimiento en sistemas modernos con múltiples núcleos.

Cada enfoque tiene sus ventajas y desventajas dependiendo del tipo de aplicación y la arquitectura del sistema. La asignación dedicada de procesadores puede ser eficaz para aplicaciones específicas donde se requiere una ejecución rápida y predecible, mientras que la programación dinámica puede ser más flexible y adaptable a cargas variables y condiciones cambiantes del sistema. La programación de subprocesos en sistemas multinúcleo, por otro lado, se centra en optimizar el uso de la memoria caché compartida para mejorar el rendimiento general del sistema.