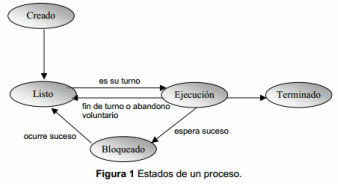
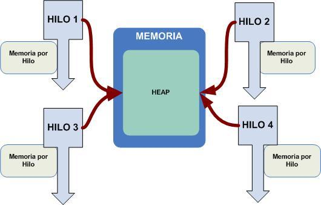
**PROGRAMACIÓN PARALELA** Jorge Pérez Flores

CICLO DE VIDA DE UN PROCESO

En un principio, un proceso no existe. En algún momento es creado. La forma de crearlo variará en función del lenguaje que se esté utilizando. Una vez creado el proceso, éste pasa al estado denominado Listo. Este estado significa que el proceso está en condiciones de hacer uso de la CPU en cuanto se le dé la oportunidad. El encargado de darle la oportunidad de usar la CPU es el denominado planificador de procesos o scheduler, que suele formar parte del SO. Como lo normal es que haya más procesos que procesadores, no todos los procesos que pueden hacer uso de la CPU en un momento determinado pueden hacerlo realmente. Esos procesos permanecen en el estado listo hasta que el planificador decide darles tiempo de CPU. Cuando el planificador decide dar tiempo de CPU a un proceso, éste pasa al estado de ejecución.

Un proceso también puede pasar de Ejecución a Listo. Esta decisión la toma el planificador. El planificador sigue algún tipo de política de planificación para asignar la CPU a los distintos procesos. Una forma bastante justa y extendida de hacerlo es mediante la asignación de rodajas de tiempo a cada uno de los procesos, de tal forma que cuando un proceso cumple su tiempo de permanencia en el procesador, éste es desalojado y pasado al estado listo. Un proceso puede pasar de Ejecución a Bloqueado cuando ha de esperar porque ocurra un determinado evento o suceso. Una vez que ocurre el evento por el que se está esperando, el proceso pasa al estado Listo. Al acto de cambiar un proceso de estado se le denomina cambio de contexto

DISPOSICIÓN EN MEMORIA DE UN PROCESO

En un Sistema Operativo tradicional, la memoria suele estar dividida en dos partes: un espacio de usuario donde suele encontrarse la mayoría de la información relativa a los procesos de usuario y un espacio del núcleo donde reside el código y las estructuras de datos propios del SO. Cuando estamos ante un SO multitarea, la información relativa a un proceso suele estar dividida entre los dos espacios (Figura 2). En el espacio de usuario se encuentra información propia del proceso tales como el código del proceso, el contador de programa, sus variables, su pila y su puntero de pila. Sin embargo, el SO necesita tener información del estado de los procesos para poder realizar apropiadamente los cambios de contexto. Esta información, que suele conocerse con el nombre de bloque de control del proceso (PCB), suele residir en el espacio del núcleo. En la Figura 3 puede verse la estructura de un proceso en el SO Unix. La información contenida en la estructura del proceso puede variar de un SO a otro, pero sustancialmente suele ser la misma.

Cuando tenemos más de un proceso, se tiene algo como lo representado en la Figura 4. En el espacio del núcleo estará el planificador de procesos que será el encargado de decidir cuándo hacer los cambios de contexto. Cuando se hace el cambio de contexto hay que recuperar la estructura del proceso que se quiere poner en el estado Ejecución y actualizar convenientemente los registros del procesador para que el nuevo proceso tome el control del mismo. Los cambios de contexto son costosos desde el punto de vista del tiempo de ejecución pues consumen un tiempo considerable. Cada proceso de los representados en la figura tiene su propio contador de programa, su propia pila, etc. Cada proceso suele tener un solo hilo de ejecución; se dice entonces que son monohilo.

¿Qué es un hilo?. De la misma manera que un Sistema Operativo puede ejecutar varios procesos al mismo tiempo bien sea por concurrencia o paralelismo, dentro de un proceso puede haber varios hilos de ejecución. Por tanto, un hilo puede definirse como cada secuencia de control dentro de un proceso que ejecuta sus instrucciones de forma independiente.En la Figura 5 puede verse cómo sobre el hardware subyacente (una o varias CPU’s) se sitúa el Sistema Operativo. Sobre éste se sitúan los procesos (Pi) que pueden ejecutarse concurrentemente y dentro de estos se ejecutan los hilos (hj) que también se pueden ejecutar de forma concurrente dentro del proceso. Es decir, tenemos concurrencia a dos niveles, una entre procesos y otra entre hilos de un mismo proceso. Si por ejemplo tenemos dos procesadores, se podrían estar ejecutando al mismo tiempo el hilo 1 del proceso 1 y el hilo 2 del proceso 3. Otra posibilidad podría ser el hilo 1 y el hilo

Los procesos son entidades pesadas. La estructura del proceso está en la parte del núcleo y, cada vez que el proceso quiere acceder a ella, tiene que hacer algún tipo de llamada al sistema, consumiendo tiempo extra de procesador. Por otra parte, los cambios de contexto entre procesos son costosos en cuanto a tiempo de computación se refiere. Por el contrario, la estructura de los hilos reside en el espacio de usuario, con lo que un hilo es una entidad ligera. Los hilos comparten la información del proceso (código, datos, etc). Si un hilo modifica una variable del proceso, el resto de hilos verán esa modificación cuando accedan a esa variable. Los cambios de contexto

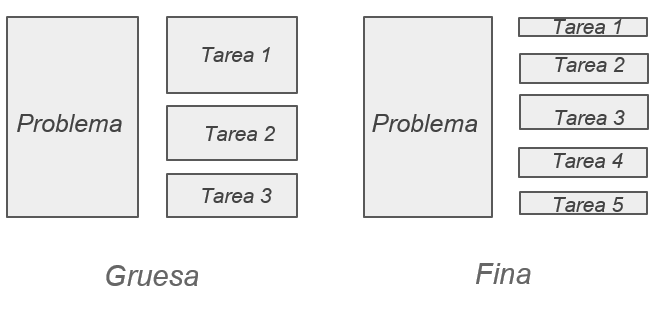
entre hilos consumen poco tiempo de procesador, de ahí su éxito.

Ventajas

* Resuelve problemas que no se podrían realizar en una sola CPU
* Resuelve problemas que no se pueden resolver en un tiempo razonable
* Permite ejecutar problemas de un orden y complejidad mayor
* Permite ejecutar código de manera más rápida (aceleración)
* Permite ejecutar en general más problemas
* Obtención de resultados en menos tiempo
* Permite la ejecución de varias instrucciones en simultáneo
* Permite dividir una tarea en partes independientes

Desventajas

* proporcional a los fallos potenciales
* Condiciones de carrera
* Múltiples procesos se encuentran en condición de Mayor consumo de energía
* Mayor dificultad a la hora de escribir programas
* Dificultad para lograr una buena sincronización y comunicación entre las tareas
* Retardos ocasionados por comunicación ente tareas
* Número de componentes usados es directamente carrera si el resultado de los mismos depende del orden de su llegada
* Si los procesos que están en condición de carrera no son correctamente sincronizados, puede producirse una corrupción de datos

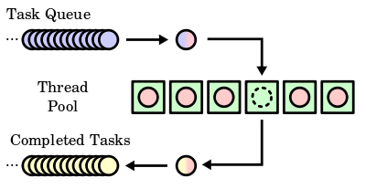
Granularidad:

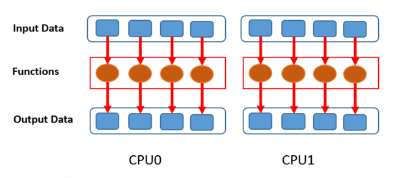
Se refiere al tamaño de cada tarea y a la independiencia de las demás tareas, se dividen en dos categorías.

Gruesa: Cantidad relativamente grande de trabajo, alta independencia entre tareas y poca necesidad de sincronización.

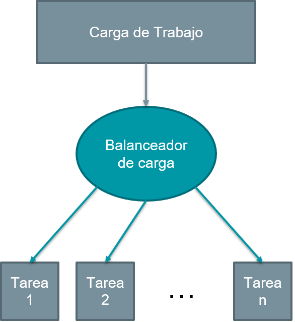
Fina: Cantidades pequeñas de trabajo, poca independencia entre tareas, y una alta demanda de sincronización.

Scheduling:

Scheduling es el proceso en el que las tareas son asignadas a los procesos o hilos, y se les da un orden de ejecución. Este puede ser especificado en el código, en tiempo de compilación o dinámicamente en tiempo de ejecución. El proceso de scheduling debe tener en cuenta la dependencia entre tareas, ya que, aunque muchas pueden ser independientes, otras pueden requerir los datos producidos por otras tareas.

Mapping:

Mapping en el proceso de asignación de procesos e hilos a unidades de procesamiento, procesadores o núcleos. Usualmente el mapping se hace por el sistema en tiempo de ejecución, aunque en ocasiones puede ser influenciado por el programador.

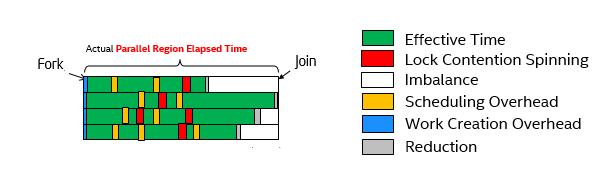
Otros conceptos

Balanceo de carga

Se refiere a la práctica de distribuir cantidades equitativas de trabajo entre las tareas, de modo que todas las tareas se mantengan ocupadas todo el tiempo.

Speedup:

Es un proceso para aumentar el rendimiento entre dos sistemas procesando el mismo problema. Es la mejora en la velocidad de ejecución de una tarea ejecutada en dos arquitecturas similares con diferentes recursos.

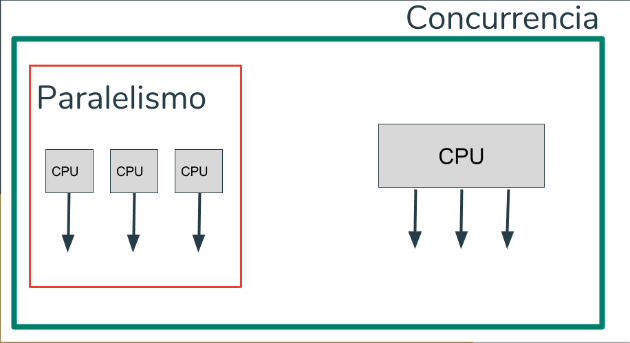
Overhead

Es la cantidad de tiempo requerido para coordinar tareas paralelas, en lugar de hacer un trabajo útil. Incluye factores como:

* Tiempo de inicio de la tarea
* Sincronización
* Comunicaciones de datos
* Sobrecarga de software impuesta por lenguajes paralelos, bibliotecas, sistema operativo, etc.
* Tiempo de terminación de la tarea

Sección crítica

Un proceso tiene un segmento de código llamado sección crítica cuando este puede modificar o leer información de memoria compartida con otros procesos. Dos procesos no pueden ejecutar su seccion critica al mismo tiempo.

Concurrencia vs Paralelismo

Concurrencia

Capacidad de operar actividades al mismo tiempo. Es decir se pueden tener varios procesos corriendo cada uno en un procesador o puede haber varios proceso que corran solo en un procesador

Paralelismo

Son muchas actividades teniendo lugar al mismo tiempo, “la cualidad o el estado de ser paralelo”. El hecho de ser paralelo implica que solo se pueden tener varios procesos corriendo cada uno en un procesador.

Computación grafica

Ciertas clases de problemas producen desequilibrios de carga incluso si los datos están distribuidos uniformemente entre las tareas

Predicción del clima y cambio climático

Uno de los primeros usos exitosos de la computación paralela fue la predicción del tiempo. La información, como la temperatura, la humedad y las precipitaciones, se ha recolectado y utilizado para predecir el clima durante más de 500 años. En 1904, el físico y meteorólogo noruego Vilhelm Bjerknes propuso un modelo de ecuaciones diferenciales para la predicción meteorológica que incluía siete variables, incluyendo temperatura, lluvia y humedad.

Análisis de imágenes multiespectro

Las imágenes satelitales consisten en grandes cantidades de datos. Por ejemplo, las imágenes Landsat 7 consta de siete tablas de datos, donde cada entrada en una tabla representa una longitud de onda magnética diferente (azul, verde, rojo o infrarrojo térmico) para un píxel de 30 metros cuadrados de la superficie de la Tierra.

Astronomía

Hay muchas aplicaciones de los supercomputadores a la astronomía, incluyendo el uso de un supercomputador para simular eventos en el futuro, o pasado, para probar teorías astronómicas. La Universidad de Minnesota Supercomputer Center simuló lo que una explosión de supernova que se origina en el borde de una gigantesca nube de gas molecular interestelar parecería 650 años después de la explosión.