Capítulo 3. Procesos concurrentes

3.1. Conceptos de programación concurrente

La computación concurrente es la simultaneidad en la ejecución de múltiples tareas interactivas. Estas tareas pueden ser un conjunto de procesos o hilos de ejecución creados por un único programa. Las tareas se pueden ejecutar en una sola unidad central de proceso (multiprogramación), en varios procesadores o en una red de computadores distribuidos. La programación concurrente está relacionada con la programación paralela, pero enfatiza más la interacción entre tareas. Así, la correcta secuencia de interacciones o comunicaciones entre los procesos y el acceso coordinado de recursos que se comparten por todos los procesos o tareas son las claves de esta disciplina. Los pioneros en este campo fueron Edsger Dijkstra, Per Brinch Hansen, y C. A. R. Hoare.

Edición concurrente

La edición concurrente consiste en que dos o más usuarios distintos editan un mismo campo de datos o un mismo documento. Algunos programas abiertos han creado módulos para evitar este problema, como sucede con Drupal.

Definición

La Ingeniería Concurrente (IC), también conocida como Paralela, es una filosofía orientada a hacer más eficiente la ingeniería, así como integrar sistemáticamente y en forma simultánea el diseño de productos y procesos. Otorgar además una organización flexible y bien estructurada, proponer redes de funciones apoyadas por tecnologías apropiadas y arquitecturas comunes de referencia (ej: computadores en red y en bases de datos).

Puede definirse como la unión de varios procedimientos que sirven para reducir los tiempos que se utilizan en el desarrollo de proyectos, teniendo en cuenta la calidad del producto, considerando desde un principio todos los elementos del ciclo de vida de un producto, desde la concepción inicial hasta su disposición final, pasando por la fabricación, la distribución y la venta. Teniendo la realización de diferentes actividades y el trabajo en diversos equipos. La ingeniería concurrente también debe de considerar los costes del ciclo de vida del producto, además de ser una gran ventaja al posicionar los productos en el mercado en un menor tiempo.

"Un sistema de trabajo donde las diferentes actividades de ingeniería en los procesos de desarrollo de producto y de proceso de producción se integran y se realizan en paralelo, siempre que sea posible, en vez de secuencialmente".

Este nuevo enfoque hacia el diseño que entrega la IC, da un gran realce al papel que juegan las personas en sus respectivos trabajos, las cuales deben estar bien instruidas.

Aunque éste no es un concepto nuevo, ha recibido recientemente cierto empuje de tecnologías de la información como Internet o algunas técnicas de Inteligencia Artificial. Específicamente, el uso de agentes de software y lenguajes para el manejo de conocimiento pueden aportar una base confiable y flexible para el desarrollo de plataformas de Ingeniería Concurrente. Respecto de la metodología de trabajo de la IC, en esencia utiliza las mismas funciones involucradas en el ciclo de desarrollo de un

producto de la forma tradicional de trabajar que es la ingeniería secuencial, a la cual reemplaza; sin embargo, la diferencia se halla en la interacción constante que se produce entre las mismas.

Objetivos

Para alcanzar los objetivos la IC utiliza una serie de principios, los cuales son empleados en un enfoque sistematizado y están relacionados con la introducción de cambios culturales, organizacionales, y tecnológicos en las compañías, a través de una serie de metodologías, técnicas y tecnologías de información.

Los objetivos globales que se persiguen con la implementación de la IC son:

1. Acortar los tiempos de desarrollo de los productos.

2. Elevar la productividad.

3. Aumentar la flexibilidad.

4. Mejor utilización de los recursos.

5. Productos de alta calidad.

6. Reducción en los costos de desarrollo de los productos.

7. Establecer conocimiento y cultura de Ingeniería Concurrente

8. Integrar los departamentos de la empresa

9. Asegurar el cumplimiento de los requerimientos y expectativas del cliente

3.2. El problema de la sección crítica

Se denomina sección crítica, en programación concurrente, a la porción de código de un programa de computador en la cual se accede a un recurso compartido (estructura de datos o dispositivo) que no debe ser accedido por más de un proceso o hilo en ejecución. La sección crítica por lo general termina en un tiempo determinado y el hilo, proceso o tarea sólo tendrá que esperar un período determinado de tiempo para entrar. Se necesita un mecanismo de sincronización en la entrada y salida de la sección crítica para asegurar la utilización en exclusiva del recurso, por ejemplo un semáforo.

El acceso concurrente se controla teniendo cuidado de las variables que se modifican dentro y fuera de la sección crítica. La sección crítica se utiliza por lo general cuando un programa multihilo actualiza múltiples variables sin un hilo de ejecución separado que lleve los cambios conflictivos a esos datos. Una situación similar, la sección crítica puede ser utilizada para asegurarse de que un recurso compartido, por ejemplo, una impresora, puede ser accedida por un solo proceso a la vez.

La manera en cómo se implementan las secciones puede variar dependiendo de los diversos sistemas operativos. Sólo un proceso puede estar en una sección crítica a la vez.

El método más común para evitar que dos procesos accedan al mismo tiempo a un recurso es el de la exclusión mutua.

En ciencias de la computación, los cierres de exclusión mutua o candados son un mecanismo de sincronización que limita el acceso a un recurso compartido por varios procesos o hilos en un ambiente de ejecución concurrente, permitiendo así la exclusión mutua.

Este mecanismo se puede ver en un ejemplo de la vida real. Supongamos un baño público, donde sólo puede entrar una persona a la vez. Una vez dentro, se emplea un cierre para evitar que entren otras personas. Si otra persona pretende usar el baño cuando está ocupado, deberá quedar esperando a que la persona que entró anteriormente termine. Si más personas llegaran, formarían una cola (del tipo FIFO) y esperarían su turno. En informática, el programador no debe asumir este tipo de comportamiento en la cola de espera.

En general hay un número de restricciones sobre los cerrojos, aunque no son las mismas en todos los sistemas. Estas son:

- Sólo el dueño de un cerrojo puede desbloquearlo
- La readquisición de un cerrojo no está permitida

Algo muy importante es que todos los procesos/hilos deben utilizar el mismo protocolo para bloquear y desbloquear los cerrojos en el acceso a los recursos, ya que si mientras dos procesos/hilos utilizan el cerrojo de forma correcta, existe otro que simplemente accede a los datos protegidos, no se garantiza la exclusión mutua y pueden darse condiciones de carrera y errores en los resultados.

Primitivas y uso

Las funciones de los cerrojos en general son tres: init(), lock() y unlock(). El cerrojo se inicializa con la función init(). Luego cada proceso/hilo debe llamar a la función lock() antes de acceder a los datos protegidos por el cierre. Al finalizar su sección crítica, el dueño del cerrojo debe desbloquearlo mediante la función unlock().

Proceso (informática)

Un proceso puede informalmente entenderse como un programa en ejecución. Formalmente un proceso es "Una unidad de actividad que se caracteriza por la ejecución de una secuencia de instrucciones, un estado actual, y un conjunto de recursos del sistema asociados".

Para entender lo que es un proceso y la diferencia entre un programa y un proceso, A. S. Tanenbaum propone la analogía "Un científico computacional con mente culinaria hornea un pastel de cumpleaños para su hija; tiene la receta para un pastel de cumpleaños y una cocina bien equipada con todos los ingredientes necesarios, harina, huevo, azúcar, leche, etcétera." Situando cada parte de la analogía se puede decir que la receta representa el programa (el algoritmo), el científico computacional es el procesador y los ingredientes son las entradas del programa. El proceso es la actividad que consiste en que el científico computacional vaya leyendo la receta, obteniendo los ingredientes y horneando el pastel.

Cada proceso tiene su contador de programa, registros y variables, aislados de otros procesos, incluso siendo el mismo programa en ejecución 2 veces. Cuando este último caso sucede, el sistema operativo usa la misma región de memoria de código, debido a que dicho código no cambiará, a menos que se ejecute una versión distinta del programa.

Los procesos son gestionados por el sistema operativo y están formados por:

- Las instrucciones de un programa destinadas a ser ejecutadas por el microprocesador.
- Su estado de ejecución en un momento dado, esto es, los valores de los registros de la unidad central de procesamiento para dicho programa.
- Su memoria de trabajo (memoria crítica), es decir, la memoria que ha reservado y sus contenidos.

Un proceso se rige en pequeñas porciones, conocidas como páginas, y cada proceso tiene su propia tabla de paginación, fungiendo como una optimización del sistema operativo ante los fallo de página.

Esta definición varía ligeramente en el caso de sistemas operativos multihilo, donde un proceso consta de uno o más hilos, la memoria de trabajo (compartida por todos los hilos) y la información de planificación. Cada hilo consta de instrucciones y estado de ejecución.

Los procesos son creados y eliminados por el sistema operativo, así como también éste se debe hacer cargo de la comunicación entre procesos, pero lo hace a petición de otros procesos (interrupción o tiempo de reloj). El mecanismo por el cual un proceso crea otro proceso se denomina bifurcación (fork). El proceso de arranque de GNU/Linux inicia con un sólo proceso (init) y después comienza a crear los hilos necesarios para tener el sistema listo para su uso. Los nuevos procesos pueden ser independientes y no compartir el espacio de memoria con el proceso que los ha creado o ser creados en el mismo espacio de memoria.

En los sistemas operativos multihilo es posible crear tanto hilos como procesos. La diferencia estriba en que un proceso solamente puede crear hilos para sí mismo y en que dichos hilos comparten toda la memoria reservada para el proceso.

Los procesos pueden ser cooperativos o independientes. Dos o más procesos pueden cooperar mediante señales de forma que uno obliga a detenerse a los otros hasta que reciban una señal para continuar.

Se usa una variable de tipo semáforo para sincronizar los procesos.

Si un proceso está esperando una señal, se suspende hasta que la señal se envíe.

Se mantiene una cola de procesos en espera en el semáforo.

La forma de elegir los procesos de la cola en espera es mediante una política first in first out.

La sincronización explícita entre procesos es un caso particular del estado "bloqueado". En este caso, el suceso que permite desbloquear un proceso no es una operación de entrada/salida, sino una señal generada a propósito por el programador desde otro proceso.

Hay cuatro eventos principales que provocan la creación de procesos:

- El arranque del sistema.
- La ejecución, desde un proceso, de una llamada al sistema para la creación de otro proceso.
- Una petición de usuario para crear un proceso.
- El inicio de un trabajo por lotes.

Los procesos pueden contener uno o más hilos, haciendo más eficiente las tareas, asimismo la complejidad de los algoritmos de sincronización, ya que podría ocurrir la condición de carrera muy a menudo, inclusive los indeseados interbloqueos.