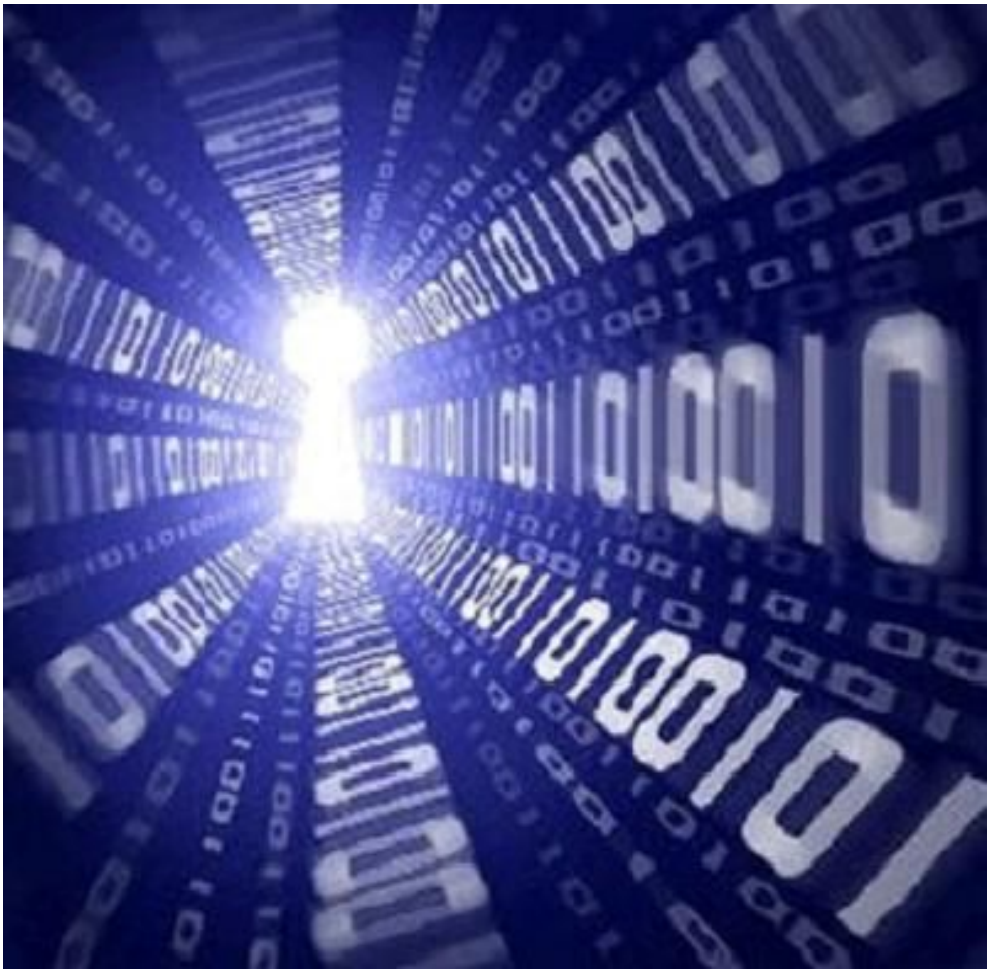
Sistemas Concurrentes y Distribuidos: Tema 1
10 de Octubre del 2020
A year to Forget

Daniel Monjas Miguélez



Índice

1. Abstracción de la PC

La Programación Concurrente es independiente de la implementación del paralelismo a bajo nivel. Se trata de abstracción que nos ayuda a programar según modelo paralelo.

La programación concurrente mejora la eficiencia de los programas y la calidad de los mismos.

- **Eficiencia en sistemas con un solo procesador:**
  - Al existir ahora varias tareas en los programas que avanzan simultáneamente, se evita la espera ociosa cualdo el programa realiza E/S.
  - Utilización del sistema por varios usuarios de forma interactiva.
- **Eficiencia en sistemas con varios procesador:**
  - Es posible repartir la ejecución de las tareas entre los distintos núcleos de un procesador.
  - Se pueden asignar tareas muy complejas a otros procesadores.
  - Para acelerar la ejecución de cálculos muy complejos.

Modelos de arquitecturas para PC: dependen de la arquitectura del procesador. Consideran una máquina virtual (multiprocesador o multicomputador) como la base común para modelar la ejecución de los procesos concurrentes. El tipo de paralelismo (monoprocesador, multiprocesador, sistema distribuido o multicomputador) afectará a la eficiencia de la ejecución pero nunca a la corrección de los programas.

Concurrencia en sistemas monoprocesador: mejor aprovechamiento de la CPU. Servicio interactivo a varios usuarios. Utilización de soluciones de diseño concurrentes. Sincronización y comunicación mediante variables compartidas.

Concurrencia en multiprocesadores o sistemas multinúcleo Los procesadores pueden compartir o no físicamente la misma memoria, pero comparten algún espacio de direcciones común. Los núcleos de los procesadores actuales comparten una memoria de acceso muy rápido y un computador podría tener además varios procesadores. La interacción entre procesos se puede implementar mediante variables ubicadas en un espacio de memoria común a todos los procesadores (variables compartidas).

Concurrencia en sistemas distribuidos: No existe memoria común, cada procesador mantiene un espacio de direcciones propio. Los procesos interaccionan transfiriéndose datos a través de una red de interconexión (paso de mensajes). Es necesario utilizar programación distribuida, que además de la concurrencia trata con otros problemas: tratamiento de fallos, transparencia, heterogeneidad, etc. Los clusters de ordenadores o de GPUS (actualmente), Internet, etc.

Sentencias atómicas y no atómicas: Una sentencia o instrucción de un proceso en un programa concurrente es atómica si siempre se ejecuta de principio a fin sin verse afectada (durante su ejecución) por las sentencias en ejecución de otros procesos del programa.

- No se verá afectada cuando el funcionamiento de dicha instrucción no dependa nunca de cómo se estén ejecutando otras instrucciones.
- El funcionamiento de una instrucción se define por su efecto en el estado de ejecución del programa justo cuando ésta acaba.
- El estado de ejecución está formado por los valores de las variables y de los registros de todos los procesos.

Abstracción: el modelo basado en el estudio de todas las posibles secuencias de ejecución entrelazadas de los procesos de un progrmaa concurrente constituye una abstracción:

- Se consideran sólo las características relevantes que determinan el resultado final del programa.
- Nos permite simplificar el análisis y estudio de los programas.

Se ignoran detalles de la computación no relevantes para obtener el resultado.

Independencia del entrono de ejecución: el entrelazamiento de instrucciones atómicas preserva la consistencia de los resultados. En caso de que no se cumpliera, sería imposible poder razonar sobre las propiedades de corrección de los programas concurrentes.

Si se cumple la hipótesis de progreso finito de los procesos, la velocidad de ejecución de cada proceso será no nula, lo cual tiene estas dos consecuencias:

- **Punto de vista global:** durante la ejecución de un programa concurrente, en cualquier momento existirá al menos un proceso preparado, es decir, eventualmente se permitirá la ejecución de algún procesos.

- **Punto de vista local:** cuando un proceso concreto de un programa concurrente comienza la ejecución de una sentencia, completará la ejecución de la sentencia en un intervalo de tiempo finito.

Estados e historias de ejecución de un programa concurrente:

- Valores de las variables del programa en un momento dado. Incluyen variables declaradas explícitamente y variables con información de estado oculta (PC, R, ...).
- Un programa concurrente comienza su ejecución en un estado inicial y los procesos van modificando el estado conforme se van ejcutando sus sentencias atómicas (producen transiciones entre dos estados de forma indivisible).

2. Exclusión mutua y sincronización

Exclusión mútua: Al conjunto de dichas secuencias comunes de instrucciones consecutivas que aparecen en el texto de varios procesos de un programa concurrente, se le denomina sección crítica (SC).

- Ocurre exclusión mutua (EM) cuando los procesos solo funcionan correctamente si, en cada instante de tiempo, hay como mucho uno de ellos ejecutando cualquier instrucción de la sección crítica.
- El entrelazamiento de las instrucciones de los procesos debe ser tal que cada secuencia de instrucciones de la SC se ejecuta como mucho por un proceso de principio a fin, sin que (durante ese tiempo) otros procesos ejecuten ninguna de esas instrucciones ni otras de la misma SC.

Condición de sincronización: es un programa concurrente, una condición de sincronización establece que todos los posibles entrelazamientos (secuencias de instrucciones atómicas de los procesos) son correctas.

- Suele ocurrir cuando, en un punto concreto de su ejecución, uno o varios procesos deben esperar a que se cumpla una determinada condición global (depende de varios procesos).

Propiedad de seguridad: condiciones que deben cumplirse en cada instante de la traza del programa.

- Requeridas en especificaciones estáticas del programa.
- Son fáciles de demostrar y para cumplirlas se suelen impedir determinadas posibles trazas.

Propiedades de vivacidad: son propiedades que deben cumplirse en algún momento futuro y no especificado del programa. Son propiedades dinámicas, más difíciles de demostrar que las propiedades de seguridad.

Axiomas y reglas de inferencia: sirven para poder llevar a cabo las demostraciones de programas, ya que cada línea de la demostración es o bien un axioma o se deriva de la anterior mediante una regla de inferencia.

1. **Axioma de la sentencia nula:** $\{P\}NULL\{P\}$: si el aserto es cierto antes de ejecutarse esta sentencia, tendrá la misma interpretación cuando la sentencia termine.
2. **Sustitución textual:**$\{P_e^x\}$ es el resultado de sustituir la expresión e en cualquier aparición de la variable x en P.
3. **Axioma de asignación:** $\{P_e^x\}x = \{P\}$: una asignación cambia solo el valor de la variable objetivo, el resto de las variables conservan los mismos valores.
4. **Regla de la consecuencia(1):** $\frac{\{P\}S\{Q\},\{Q\}\rightarrow\{R\}}{\{P\}S\{R\}}$
5. **Regla de la composición(2):** $\frac{\{R\}\rightarrow\{P\},\{P\}S\{Q\}}{\{R\}S\{Q\}}$
6. **Regla de la composición:** $\frac{\{P\}S_1\{Q\},\{Q\}S_2\{R\}}{\{P\}S_1;S_2\{R\}}$ para obtener la pre y pos-condición de 2 sentencias juntas.
7. **Regla del IF:** $\frac{\{P\}\wedge\{B\}S_1\{Q\},\{P\wedge\neg B\}S_2\{Q\}}{\{P\}if\ B\ then\ S_1\ else\ S_2\ fi\{Q\}}$
8. **Regla de la iteración:** $\frac{\{I\wedge B\}S\{I\}}{\{I\}while\ B\ do\ S\ enddo\{I\wedge\neg B\}}$ podrá iterar un número arbitrario de veces (incluso 0); el invariante I se satisface antes y después de cada iteración.
9. **Regla conjunción:** $\frac{\{P\}S\{Q_1\},\{P\}S\{Q_2\}}{\{P\}S\{Q_1\wedge Q_2\}}$
10. **Regla disyunción:** $\frac{\{P_1\}S\{Q\},\{P_2\}S\{Q\}}{\{P_1\vee P_2\}S\{Q\}}$
11. **Regla conjunción inversa:** $\frac{\{P\}S\{Q_1\wedge Q_2\}}{\{P\}S\{Q_i\}}$
12. **Regla disyunción inversa:** $\frac{\{P_1\vee P_2\}S\{Q\}}{\{P_i\}S\{Q\}}$

3. Glosario

- **Programa Secuencial:** declaraciones de datos + conjunto de instrucciones ejecutables en secuencia.

- **Programa Concurrente:** conjunto de programas secuenciales que se pueden ejecutar lógicamente en paralelo.
- **Proceso:** ejecución de un programa secuencial.
- **Concurrencia:** potencial para la ejecución paralela de código: entremezclamiento de instrucciones de varios programas (real o virtual).
- **Programación Concurrente:** conjunto de notaciones y técnicas de programación utilizadas para expresar el paralelismo potencial y resolver problemas de sincronización y comunicación.
- **Programación paralela:** enfocada a acelerar la resolución de problemas de eficiencia de los cálculos mediante el mejor aprovechamiento de la capacidad de proceso paralelo del hardware.
- **Programación distribuida:** su objetivo es hacer que varios componentes software localizados en distintos ordenadores trabajen juntos.
- **Programación de tiempo real:** resuelve la programación de sistemas que funcionan continuamente, reciben entradas/salidas desde componentes hardware y reaccionan frente a éstas, se han de cumplir plazos estrictos de tiempo para realizar las areas del programa.
- **Multiprogramación:** el sistema operativo gestiona cómo múltiples procesos se reparten los ciclos del procesador.
- **Hipótesis de progreso finito:** no se puede hacer ninguna suposición acerca de las velocidades absolutas/relativas de ejecución de los procesos, salvo que es mayor que cero. Un programa concurrente se entiende sólo en base a sus componentes (procesos) y sus interacciones, sin tener en cuenta el entorno de ejecución.
- **Historia o traza de un programa concurrente:** secuencia de estados, producida por una secuencia de entrelazamiento.
- **Grafo de sincronización:** es un grafo dirigido acíclico (DAG) donde cada nodo representa una secuencia de sentencias del programa (actividad).
- **fork:** sentencia que especifica que la rutina nombrada puede comenzar su ejecución, al mismo tiempo que comienza la sentencia siguiente (bifurcación).
- **join:** sentencia que espera la terminación de la rutina nombrada, antes de comenzar la sentencia siguiente (unión).

- **Condiciño de sincronización:** restricción en el orden en que se pueden entremezclar las instrucciones que generan los procesos de un programa.
- **Exclusión mutua:** se da en secuencias finitas de instrucciones del código de un programa, quehan de ejecutarse de principio a fin por un único proceso, que no puede ser desplazado mientras ejecuta esta sección crítica de instrucciones.
- **Propiedad de un programa concurrente:** algo que se puede afirmar del programa que es cierto para todas las posibles secuencias de entrelazamiento (trazas del programa).
- **Verificación: enfoque axiomático:** se define un sistema lógico forma que permite establecer propiedades de programas en base a axiomas y reglas de inferencia.
- **Invariante global:** se trata de un predicado que referencia variables globales del programa que se demuestra cierto en el estado inicial de cada proceso y se mantiene cierto ante cualquier asignación dentro de los procesos.
- **Condición de invariante global (IG):** dado un aserto I definido a partir de las variables compartidas entre los procesos de un programa concurrente.