SCD Temas 1 y 2

Aurelia Nogueras

Contents

Tema 1: Introducción

Conceptos básicos

- Programa secuencial: se ejecuta en secuencia.
- Programa concurrente: se puede ejecutar en paralelo.
- Proceso: ejecución de un programa secuencial.
- Concurrencia: describe el potencial para la ejecución paralela.
- Programación concurrente: técnicas y notaciones para paralelismo potencial y resolver problemas de sincronización y comunicación.
- Programación paralela: acelera la resolución de problemas mediante el procesamiento en paralelo.
- Programación distribuida: varios componentes en diferentes localizaciones trabajando juntos.
- Programación de tiempo real: sistemas que están funcionando continuamente con restricciones estrictas en la respuesta temporal.

La programación concurrente nos permite mejorar la eficiencia y la calidad.

Modelo abstracto y consideraciones sobre el hardware

- Concurrencia en sistemas monoprocesador: paralelismo virtual. Múltiples procesos se reparten los ciclos de CPU (multiprogramación). Sincronicación y comunicación mediante variables compartidas.
- Concurrencia en multiprocesadores de memoria compartida: los procesadores comparten un espacio de direcciones (pueden o no compartir memoria). Pueden usar variables compartidas en dicho espacio de direcciones.
- Concurrencia en sistemas distribuidos: los procesadores interactúan a través de una red de interconexión (no hay memoria común).

Dentro del modelo abstracto de concurrenia, podemos distinguir sentencias atómicas y no atómicas.

- Sentencia atómica: se ejecuta de principio a fin sin verse afectada por otras sentencias. Por ejemplo: leer una celda de memoria y cargar su valor en un registro, incrementar el valor de un registro o escribir el valor de un registro en una celda. El estado final está determinado al no depender de otras instrucciones.
- Sentencias no atómicas: la mayoría de las sentencias son no atómicas. En ellas hay indeterminación del estado final. Las interfoliaciones son las posibles mezclas de las sentencias atómicas.
- El entrelazamiento preserva la consistencia.
- No se hacen suposiciones temporales, salvo que el tiempo es mayor que cero y el progreso es finito.
- Estado e historia de un programa concurrente.

- Sistemas estáticos y dinámicos: según puedan o no variar el número de procesos durante la ejecución.
- Grafo de sincronización: grafo dirigido acíclico. Muestra las restricciones.

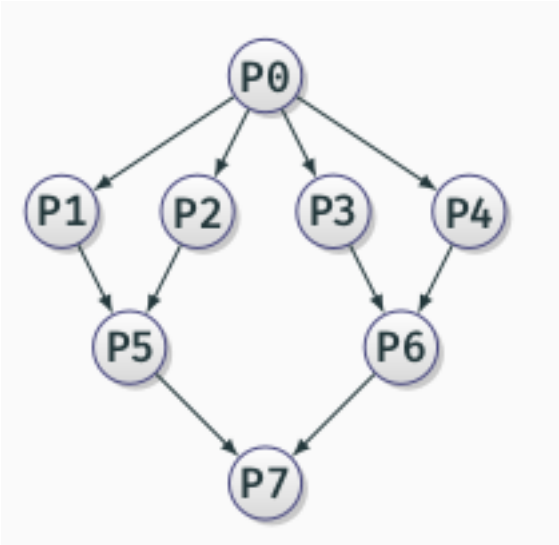

Figure 1: Grafo

- Definición estática de procesos: se usa la clave **process**. Las variables compartidas se inicializan antes de empezar ningún proceso. También están los vectores de procesos, que se diferencian en el valor de una constante.
- **Fork-Join**: fork indica que la rutina nombrada puede comenzar su ejecución al tiempo que comienza la sentencia siguiente (bifurcación). Por otro lado, join espera la terminación de la rutina nombrada antes de proseguir con la sentencia siguiente. Es práctica y potente, pero de difícil comprensión al no estar estructurada.
- **Cobegin-coend**: los procesos están estructurados. Las sentencias en un bloque cobegin-coend comienzan su ejecución todas a la vez. Es menos potente que fork-join, pero más comprensible.

Exclusión mutua y sincronización

Algunas de las interfoliaciones posibles en un conjunto de procesos no son válidas para el funcionamiento requerido de nuestro programa. Esto implica que hay una condición de sincronización o restricción sobre el orden de mezcla. Un caso particular de sincronización es la exclusión mutua, que son secuencias finitas de instrucciones que deben ejecutarse de principio a fin por un único proceso sin que otro proceso esté ejecutándolas al mismo tiempo.

- Exclusión mutua: a las instrucciones restringidas se les denomina sección crítica. La exclusión mutua tiene lugar cuando, en cada instante de tiempo, solo hay un proceso ejecutando cualquier instrucción de la sección crítica (de principio a fin).
- Condición de sincronización: no son correctas todas las posibles interfoliaciones. Por ejemplo, en el problema del productor-consumidor.

Propiedades de los sistemas concurrentes

Las propiedades son los atributos ciertos para todas las secuencias de interfoliación. Hay dos tipos: de seguridad y de vivacidad.

- Propiedades de seguridad (safety): deben cumplirse en cada instante, como la exlusión mutua, la ausencia de interbloqueo (los procesos esperan algo que nunca sucederá)...
- Propiedades de vivacidad (liveness): deben cumplirse eventualmente. Son dinámicas, como la ausencia de inanición (un proceso no puede ser indefinidamente pospuesto) o la equidad (justicia relativa respecto a los procesos).

Verificación de programas concurrentes

Para probar que un programa cumple una propiedad, podemos enfocarlo de distintas formas:

- Posibilidad: realizando ejecuciones. No obstante, solo consideramos un número limitado de historias.
- Enfoque operacional: análisis de todos los posibles casos. El problema es que puede haber un gran número de interfoliaciones.
- Enfoque axiomático: a partir de fórmulas lógicas se prueba la verificación deseada. El invariante global es el predicado que referencia variables globales siendo cierto en el estado inicial y manteniéndose ante cualquier asignación. En el productor-consumidor, sería: `consumidos <= producidos <= consumidos + 1`.

Fuentes

Las fuentes de esta exposición son las siguientes:

Temario SCD: En PRADO, dentro de la parte teórica de la asignatura.