Programación concurrente

Clase 1:

Temas:

Conceptos básicos
Concurrencia por memoria compartida
Concurrencia por pasaje de mensajes (MP)
Lenguajes que soportan concurrencia
Introducción a la programación paralela

El parcial se divide en 2 grandes temas:

Memoria Compartida
Memoria Distribuida - Pasaje de Mensajes

Introducción:

¿Qué es la concurrencia? Simultaneidad, capacidad de ejecutar múltiples eventos simultáneamente. Estos eventos pueden interactuar entre si, por lo que hay que evitar conflictos entre ellos.
Un navegador web accediendo a una pagina mientras se atiende al usuario, es un ejemplo de concurrencia, en general todo aspecto en el que se vea mas de un evento funcionando a la vez puede servir como ejemplo.

Todo el mundo es concurrente, un gran numero de células, que trabajan simultáneamente en un por ejemplo, animal.

Un proceso secuencial es raro de encontrar en el mundo real.

Imaginar el siguiente problema: Desplegar cada 3 segundos un cartel rojo y cada 5 segundos otro cartel.
Es complejo implementar una solución secuencial. Dado la asimentria del tiempo de cada cartel.

Solución: Imaginarlo con concurrencia, por cada cartel un “Programa distinto” que se ejecuta de forma simultanea con el otro.

Con concurrencia se obtienen aplicaciones con una estructura mas natural, con un mejor rendimiento, y para obtener sistemas distribuidos.

Objetivo de la concurrencia:

Ajustar el modelo de arquitectura de hardware y software al problema del mundo real que se busca resolver.

Incrementar el rendimiento, lo que seria el tiempo de ejecución, aparte de ahorrar energía.

Ventajas:

Mas velocidad de ejecución.

Mejor utilización de la CPU.

Nucleo(Core): Mientras mas núcleos, mayor cantidad de instrucciones paralelas que se pueden ejecutar.

Procesamiento secuencial: Tareas una a la vez, agarro una tarea la hago, agarro la siguiente y la hago. Establece un orden de pasos rígido.

Procesamiento concurrente: Se sigue teniendo una tarea a la vez, pero el sistema se encarga de ir otorgándole tiempo a cada programa distinto, siendo algo que ocurre de forma simultanea, pero no paralelamente. Esto permite ejecutar varios programas a la vez incluso teniendo un solo nucleo en el procesador.

Procesamiento paralelo (Se le llama solucion paralela): Una forma concurrente pero incluyendo paralelismo, lo que permite que se ejecuten directamente al mismo tiempo. Se requiere del hardware apropiado, con una cantidad de núcleos que pueda soportar el paralelismo requerido.

Cambios de contexto: Los procesos (Programas) trabajan con los mismos registros, por lo que se requiere de cambios de contexto para que no ocurran conflictos por Datos erróneos. Esto es una dificultad aplicada a los procesamientos que son concurrentes pero no paralelos.

Dificultad del paralelismo:

Distribución de la carga del trabajo
Necesidad de compartir recursos
Necesidad de comunicarse (Esperar puntos claves)

Puede existir el caso en el que haya 2 núcleos por ejemplo, y 5 tareas simultáneas, en ese caso tambien se aplican cambios de contexto.

Un programa concurrente consiste en múltiples procesos que trabajan “Simultaneamente” que juntos resuelven un mismo problema con recursos compartidos.
Estos procesos a veces cooperan y otras veces compiten.

Un programa concurrente puede tener P procesos habilitados y un sistema concurrente puede disponer de UP (Unidades de procesamiento)
Si UP es mayor a 1 se le puede decir sistema paralelo, en caso contrario es solo un sistema concurrente.

Los procesos trabajan de una de estas dos formas distintas:

Competencia: La competencia entre procesos es tipo en sistemas operativos y redes, consiste en competir por los recursos.

Cooperación: La cooperación consiste en colaborar para resolver una tarea común.

Cada proceso tiene su propio espacio de direcciones y recursos.

Proceso liviano: tambien llamado “Hilo”, tiene su propio contador de programa y pila de ejecucion, pero no controla el “contexto pesado” (Ej: Tablas de paginas). Se dice que el hilo tiene un contexto mas liviano (Que en vez de pesar 5 Kb, pesa 5 Bytes)
Estos hilos comparten el mismo espacio de direcciones y recursos.

¿Como se comunican los procesos entre si?

Existen dos métodos que pueden tener los procesos para comunicarse, por memoria compartida, o por memoria Distribuida.

Memoria compartida: Se asume que la arquitectura posee de alguna forma dentro de los núcleos, un espacio común de comunicación, que es usado por los procesos para comunicarse entre ellos. En este caso puede haber riesgos de accesos indebidos, por lo que hay que tener claro control sobre ellos.

Memoria distribuida: Cada uno tiene sus propia memoria, por lo que tienen de comunicarse por medio de mensajes.

Sincronización: Posesion de información acerca de otro proceso para coordinar actividades, estos pueden sincronizarse por los siguientes métodos:

Por exclusión mutua: Un área del programa que incluye recursos compartidos, y que solo es accedida por un proceso a la vez. Esto se aplica principalmente a los recursos que se quieren compartir.

Por condición: Permite bloquear la ejecucion de un proceso hasta que se cumpla una condición dada. “Dormir un proceso hasta que ocurra algo”

Interferencia: Un proceso toma una acción que invalida las suposiciones hechas por otro proceso.

Ejemplo: Tenemos dos procesos, ambos usan una variable compartida Y, se ejecuta una instrucción if (Y =! 0), da true y se mete dentro del if, pero justo en ese momento se cambia al otro programa cuya primera instrucción es Y = 0. Estarías entrando a un if que no tuviste que entrar bajo la lógica del primer programa.

Administracion de recursos compartidos: Como se asignan los recursos, que métodos se toman para acceder a los mismos, como se bloquean y liberan, en general se busca tener control de estos y no dejar que el sistema operativo decide, ya que esto puede provocar interferencias.

El concepto de inanicion tambien esta presente en la concurrencia. Puede pasar que haya procesos que no realicen mucho trabajo, quedando en inanición.

Deadlock: Un problema que puede haber entre procesos. Consiste en un bloqueo que imposibilita que el programa concurrente siga ejecutándose ya que uno o mas procesos quedan a la espera de algo que nunca obtendran.

Un lenguaje concurrente debe proveer lo siguiente:

Indicar las tareas o procesos que pueden ejecutar se concurrentemente.
Mecanismos de sincronización.
Mecanismos de comunicación entre procesos.

No determinismo: Termino que aplica al hecho de que una ejecucion de un programa concurrente no necesariamente es idéntica a otra ejecución del mismo programa.

3 clases de intrucciones: Asignación, alternativa (Decision), e iteracion.

Intrucciones nuevas:

Skip: equivalente al NOP, no hace nada.

Sentencias de alternativa múltiple: un if no determinista, parece un if else. Pero no lo es, ya que varias condiciones pueden ser verdaderas, en ese caso, se elige una de esas y se ejecuta, por eso es no determinista.

Sentencias de alternativa iterativa múltiple: Es como el anterior, pero se va ejecutando una y otra vez hasta que no quede ninguna condición verdadera.

Co: Esta sentencia permite incluir varias instrucciones que se ejecutan simultaneamente, y no continua con la ejecución del programa hasta que cada instrucción se haya terminado de ejecutar.

Process: Sentencia que se usa para declarar un proceso, que sirve como un pequeño programa dentro de la ejecución concurrente de procesos.

Acción atomica: Realiza una transformación de estado indivisible.
Cada proceso ejecuta un conjunto de sentencias, cada una implementada por una o mas acciones atómicas.

Es decir, la ejecución de un programa concurrente es un intercalado de acciones atómicas ejecutadas por procesos individuales.

Historia (Trace): es una ejecución particular de todo el conjunto de posibles ejecuciones que puede tener un programa concurrente.

Algunas historias son válidas y otras no.

Una acción atomica por grano fino debe implementarse por medio de hardware. (Este termino se asocia a instrucciones muy fundamentales, como por ejemplo un MOVE típico)

Por ejemplo, una asignación del tipo A=B NO es atomica de grano fino, debido a que está asignación implica un par de pasos que realiza el sistema por detrás, estos pasos que el sistema realiza SI son acciones atómicas de grano fino.

Await(B) es una instrucción que permite suspender la ejecución hasta que la condición B se cumpla, y tras eso ejecutar un conjunto de instrucciones.

Fairness: Concepto relacionado al hecho de que cada proceso tenga la chance de avanzar sin importar lo que hagan los demás.