Clase 1-intro

miércoles, 14 de agosto de 2024 16:11

Concurrencia: Concepto de software no restringido a una arquitectura particular de hardware ni a un nro específico de procesadores.
Concurrente es el PROGRAMA. Dividir lo que quiero hacer en partes

Cosas que se ejecutan en simuláneo para resolver un mismo problema/usar un recurso compartido

- Soluciones secuenciales: nos fuerzan a un orden estricto temporal
- Concurrentes: una parte del tiempo se dedica a cada tarea. Se aprovechan los tiempos muertos. Concurrencia sin paralelismo
- Paralela: paralelismo de hardware, todos los cores trabajan en simultáneo.

CONCURRENCIA ⇒ Concepto de software no restringido a una arquitectura particular de hardware ni a un número determinado de procesadores.
Especificar la concurrencia implica especificar los ***procesos concurrentes***, su ***comunicación*** y su ***sincronización***.

PARALELISMO ⇒ Se asocia con la ejecución concurrente en múltiples procesadores con el objetivo principal de reducir el tiempo de ejecución.

Paralela y concurrente son ambos concurrentes

- Menos esfuerzo individual: se gasta menos cada core pq se usa menos
- En ejecuciones paralelas se tarda mucho menos
- Hay que distribuir bien la carga de trabajo (si le doy 100 a uno y 1 a otro la cago)
- Compartir recursos bien
- Esperarse en momentos clave
- Comunicarse
- Tratar bien las fallas

Proceso: elemento computacial en el que se divide mi coso. Programa concurrene compuesto por procesos

Sistema concurrente: Un programa concurrente puede tener N procesos y un sistema concurrente M elementos de procesamiento

Procesos se pueden comportar de 3 formas:

- Independientemente: cada proceso trabaja por su cuenta, no comparte info, no se espera, etc
- Competencia: procesos compiten para usar recurso compartido. EJ: impresora en empresa. Sincronización por exclusión mutua
- Cooperativamente: se divide trabajo en varios procesos y la suma de ellos hace el trabajo. Sincronización por ????

No determinismo: puede ser que al correr el programa dos veces con las mismas entradas me de algo distinto. En programas secuenciales esto ni ahi pasa.
Es difícil debuguear pq puede ser que cada print me de algo istinto

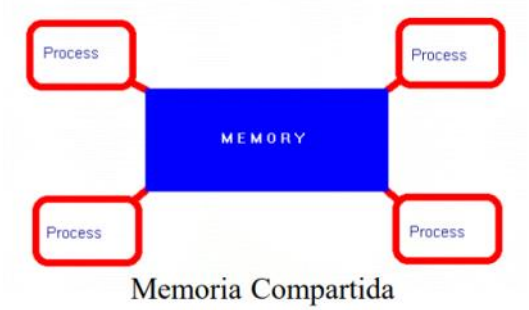
Por qué es importante concurrente?

- Hay cosas que naturalmente son concurrentes
- Mejor respuesta
- Sistemas distribuidos--> única solución. Ejemplo venta de entradas online.
- Mejoras de arquitectura: multicore, redes de conexión.

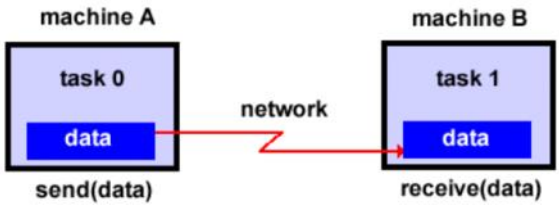
Procesos: elemento con su propio espacio de direcciones y recursos
Proceso liviano/hilo: proceso chikito que comarte espacio de direcciones del proceso que lo generó pero tiene su propio pc y pila pero no controla el contexto pesado (ej: tablas de páginas)
Contexto del proceso es mucho más chico que el del proceso que lo generó
Context switch es mucho más liviano que cmbiar de proceso pq ya está adentro del proceso. Threads pueden compartir cositas porque tienen = espacio de direcciones

Comunicación entre procesos: protocolo para compartir información

- Pasaje de mensajes: protocolo para hacer comunicación a mano. Canal físico o simulado (virtual). Se usa en general para memoria distribuida
  - Más difícil que funcione
  - Más difícil cagarla
- Memoria compartida (espacio de direcciones compartida): lo puedo hacer en un solo core, en multicore, pero NO en clusters porque la memoria es distinta.
  - Más facil que cometa errores pero + fácil de ejecutar
  - Hay que bloquear y liberar para no cagarla



Memoria Compartida



Pasaje de Mensajes

Sincronización: saber qué están haciendo los otros procesos para coordinar las actividades entre ellos.

- Por exclusión mutua: entramos de a uno a la vez. Ejemplo: impresora.
  - Se usa cuando hay secciones críticas. No puedo permitir que dos procesos lo usen en simultáneo
- Por condición: tiene que pasar algo para que yo pueda acceder al dato. Que esté libre no es suficiente

A veces se usan los dos

Interferencia: un proceso ma una acción que invalida lo que supone otro proceso. Es algo malo. Ejemplo que me modifiquen un dato que pensé que no me iban a tocar

Manejo de recursos compartidos:

- Es un problema sobre todo cuando no hay un administrador
- Hay que tener cuidado con el deadlock, que no se nos bloquee todo
- Tiene que ser justo el acceso, equilibrio.
- Que no haya inanición ni overloading

Deadlock: si ambos se quedan esperando que el otro libere un recurso compartido. Que se bloqueen los procesos sin poder contnuar. Se quedan de por vida tiesos.

Para que haya deadlock tiene que haber (sí o sí, pero puede ser que se den todas y no se de deadlock):

- **Recursos reusables serialmene:** los procesos comparten recrsos que se usan con exclusión mutua
- **Adquisición incremental:** necesito al menos dos recursos compartidos para hacer cositas pero uso primero uno (me lo quedo) y después el otro. Para continuar necesito ambas cosas pero las agarro de a una, no puedo seguir hasta no conseguir lo que tiene el otro y viceversa
- **No-preemption:** Los procesos tienen igual fuerza. Uno no le puede sacar al otro
- **Espera ciclica:** cadena circular de procesos. Cada uno tiene un recurso que el otro necesita y necesita algo de otro que el otro no larga

Lenguajes de programación concurrentes

- Indicar las tareas o procesos que se pueden ejecutar concurrentemente
- Mecanismos de sincroniación
- Comnicación

Problemas

- Más tiempo de desarrollo porque es más complejo
- Puede haber menos performance por overhead de context switch
- No determinismo
- Adaptar el software al hardware que estoy usando