TEMA 10
PROCESOS Y PROCESADORES EN SISTEMAS DISTRIBUIDOS

10.1 INTRODUCCION A LOS HILOS (THREADS): Los hilos son como mini procesos que se ejecutan de forma estrictamente secuencial. Tiene su propio contador de programa y una pila para llevar un registro de su posición. Los hilos comparten la cpu igual que los procesos: (secuencial mente en tiempo compartido) en un multiprocesador se pueden ejecutar realmente en paralelo. Los hilos pueden crear hilos hijos. Mientras un hilo esta bloqueado se puede ejecutar otro hilo del mismo proceso. Los distintos hilos de un proceso pueden compartir un espacio de direcciones, el conjunto de un archivo abierto, cronómetros, señales, etc. los hilos pueden tener distintos estados en ejecución bloqueado, listo, terminado.

10.2 USO DE HILOS: Los hilos permiten la combinación del paralelismo con la ejecución secuencial y el bloqueo de las llamadas al sistema.
Posibles métodos de organización para muchos hilos de ejecución:
Modelo servidor / trabajador: Un hilo, el **servidor**, lee las solicitudes de trabajo en el buzón del sistema. Elige a un **hilo trabajador** inactivo (bloqueado) y envía la solicitud, despertándolo. El h. Trabajador verifica si puede satisfacer la solicitud y lo hace. Los hilos ganan un desempeño considerable pero cada uno de ellos se programa en forma secuencial.
Modelo de equipo: todos los hilos son iguales y c / u obtiene y procesa sus propias solicitudes. No hay servidor. Hay una cola de trabajo que contiene todos los trabajos pendientes.
Modelo de entubamiento: El primer hilo genera datos y los transfiere al siguiente para su procesamiento. Los datos pasan de hilo en hilo y en cada etapa se lleva a cabo cierto procesamiento.
Un programa diseñado adecuadamente y que utilice hilos debe funcionar bien en una única CPU y en un multiprocesador.

10.3 ASPECTOS DEL DISEÑO DE UN PAQUETE DE HILOS: un conjunto de primitivas relacionadas con los hilos disponibles para los usuarios se llama un "PAQUETE DE HILOS".
Se tiene hilos estáticos y dinámicos.
En un **diseño estático:** Se elige el n° de hilos al escribir el programa o durante su compilación. C / u de ellos tiene asociada una pila fija. Se logra simplicidad pero también inflexibilidad.
En un **diseño dinámico:** Se permite la creación y destrucción de hilos durante la ejecución. Los hilos comparten una memoria común y el acceso se controla mediante *regiones criticas.*

10.4 IMPLANTACION DE UN PAQUETE DE HILOS: Un paquete de hilos se puede implantar en el *espacio del usuario o del núcleo.*
Implantación de un paquete de hilos en el espacio del usuario: El nucleo no sabe de su existencia y maneja procesos con un único hilo. El intercambio de hilos es mas rápido que si se usara señalamiento al núcleo, cada proceso puede tener su propio logaritmo de planificación de hilos.
Implantación de un paquete de hilos en el espacio del núcleo: para cada proceso el núcleo tiene una tabla con una entrada para cada hilo que contiene información del hilo o un hilo de otro proceso.
Los paquetes de hilos a nivel de usuario tienen el problema que si un hilo se ejecuta no se puede ejecutar ningún otro de ese proceso.

10.5 HILOS Y RPC: en los sistemas distribuidos es común utilizar hilos y RPC. Con un hilos servidor "S" y un hilo cliente "C", para llamar al servidor, el cliente coloca sus argumentos en la pila compartida, hace un señalamiento al núcleo, el núcleo detecta y deduce que es una llamada local, inicia el hilo cliente, el ejecutar el procedimiento del servidor. La llamada se efectúa porque los argumentos están en su lugar y no es necesario su copiado u ordenamiento.
La RPC local se realiza más rápido de esta forma.

10.6 MODELOS DE SISTEMAS: En un sistema distribuido, con varios procesadores, es importante el diseño de cómo se los va a utilizar. Los procesadores distribuidos se pueden organizar de varias formas:

- Modelo de estación de trabajo
- Modelo de pila de procesadores
- Modelo híbrido

10.7 EL MODELO DE ESTACIÓN DE TRABAJO. El sistema consta de estaciones de trabajo (PC) dispersas conectadas entre si mediante una red de área local (LAN). Pueden contar o no con disco rígido en c / u de ellas. Los usuarios tienen una cantidad fija de poder de cómputo exclusivo y un alto grado de autonomía para asignar recursos de estación de trabajo.

10.8 USO DE ESTACIONES DE TRABAJO INACTIVAS: se ordena remotamente la ejecución de procesos en estaciones de trabajo inactivos. Una estación de trabajo estará inactiva cuando nadie toca el ratón o teclado por varios minutos y no se ejecutan algún proceso iniciado por el usuario.
Existen algoritmos para localizar las estaciones de trabajo inactivas:

- *Algoritmos controlados por el servidor:* Cuando una estación de trabajo esta inactiva se convierte en un servidor potencial. Debe anunciar su disponibilidad en la red.
- *Algoritmos controlados por el cliente:* El cliente transmite una solicitud con sus resultados. Si regresa las respuestas se elige una estación inactiva y se la configura.

Para ejecutar un proceso en una estación remota seleccionada se debe lograr el desplazamiento del código y la configuración del proceso remoto para que vea el mismo ambiente que en el área local, y se ejecuta de la misma forma. Si regresa el poseedor de la máquina se elimina el proceso intruso abruptamente u ordenadamente, o bien se podría emigrar el proceso a otra estación.

10.9 EL MODELO DE LA PILA DE PROCESADORES: Se dispone de un conjunto de cpu que se pueden asignar dinámicamente a los usuarios según la demanda. Los usuarios no disponen de estaciones de trabajo sino de terminales graficas de alto rendimiento.
El argumento para la centralización del poder de cómputo como una pila de procesadores proviene de la teoría de colas. Se reúnen todas las CPU y se forma una sola pila de procesadores, y tenemos un solo sistema de colas en vez de "n" colas ejecutándose en paralelo.
Este modelo de pila es más eficiente que el modelo de búsqueda de estaciones inactivas.
También existe el **modelo híbrido** que consta de estaciones de trabajo y una pila de procesadores.

10.10 ASIGNACION DE PROCESADORES: Son necesarios algoritmos para decidir cual proceso hay que ejecutar y en que maquina.

- Para el modelo de estaciones de trabajo: Decidir cuando ejecutar el proceso de manera local y cuando buscar una estación inactiva. Para el modelo de la pila de procesadores: Decidir donde ejecutar cada nuevo proceso.

10.11 MODELOS DE ASIGNACIÓN. Se piensa que todas las máquinas son idénticas (o compartibles con el código), difieren a lo sumo en la velocidad, y que cada procesador se puede comunicar con los demás.
Estrategias de asignación de procesadores:

- **No migratorias:** si se colocó un proceso en una maquina permanece ahí hasta que termina.
- **Migratorias:** Un proceso se puede trasladar aunque haya iniciado su ejecución. Permiten un mejor balance de la carga pero son más complejas.

Los algoritmos de asignación intentan optimizar algo:

- **Uso de las cpu**
- **Tiempo promedio de respuesta**
- **Tasa de respuesta**

10.12 ASPECTOS DEL DISEÑO DE ALGORITMOS DE ASIGNACION DE PROCESADORES:

- **Algoritmos deterministas vs. Heurísticas.** Los deterministas son adecuados cuando se sabe de antemano todo sobre el comportamiento de los procesos. Los heurísticos cuando la carga es imprescindible.
- **Algoritmos centralizados vs. Distribuidos.** Los centralizados reúnen toda la información en un lugar y toman una mejor decisión, pero la máquina central se puede sobrecargar.
- **Algoritmos óptimos vs. Suboptimos.** Los óptimos consumen más recursos que los subóptimos. En los sistemas reales y buscan solución subóptimos, heurísticos y distribuidos.
- **Algoritmos locales vs. Globales.** Cuando se crea un proceso se divide si se va a ejecutar en la máquina que lo generó o en otra. "POLITICA DE TRANSFERENCIA". Los locales son sencilla pero no óptimos y los globales son mejores pero consumen muchos recursos.
- **Algoritmos iniciados por el emisor vs. Iniciados por el receptor.** Cuando una máquina se deshace de de un proceso debe decidir donde enviarlo: "POLITICA DE LOCALIZACIÓN". Se necesita de la información de la carga en todas las partes, obteniéndola de un emisor sobrecargado que busca una máquina inactiva o de un receptor desocupado que busca trabajo.

10.13 ASPECTOS DE LA IMPLANTACION DE ALGORITMOS DE ASIGNACION DE PROCESADORES: los algoritmos suponen que las maquinas conocen su propia carga y pueden informar su estado.

- **La *medición de la carga* no es tan sencilla**. No es sencilla. Algunos métodos: *contar el numero de procesos o solo los procesos en ejecución o listos, *Ó medir el tiempo que la cpu esta ocupada.
- **El *costo excesivo* en consumo de recursos** para recolectar medidas y desplazar procesos, ya que se debería considerar el tiempo de CPU, ancho de banda y el uso de memoria.
- **La complejidad del software:** desempeño, correctéz y robustez del sistema.
- **Estabilidad del sistema:** las máquinas ejecutan sus algoritmos en forma asincrónica por lo que el sistema nunca se equilibra.

10.14 EJEMPLOS DE ALGORITMOS DE ASIGNACION DE PROCESADORES:

- **Un algoritmo determinista según la teoría de graficas.** Precisa información de antemano como los requerimientos de cpu y de memoria de los procesos; si el núcleo de proceso supera al número de CPU, se asignará varios procesos a la misma CPU, la asignación debe minimizar el tráfico en la red.
- **Un algoritmo centralizado.** No necesita información de antemano, es un algoritmo arriba-abajo porque un coordinador mantiene la tabla de usos. No se minimiza el uso de CPU, se procura otorgar a cada usuario una parte justa del poder de cómputo. Cuando la máquina donde se crea un proceso decide que lo va a ejecutar en otra, pide al coordinador que le asigne un procesador.
- **Un algoritmo jerárquico.** En el centralizado el modo central se convierte en un cuello de botella y en un único punto de fallo. Se organizan los procesadores en jerarquías lógicas independientes de las estructuras. Para cada grupo de máquinas hay una máquina administradora que registra las máquinas ocupadas y las inactivas. En caso de falla de un equipo, lo puede reemplazar un subordinado.
- **Un algoritmo distribuido heuristico (eager).** Al crearse un proceso, la maquina donde se origina envía mensajes de prueba a una maquina para preguntar si su carga es menor; si responde que sí, el proceso se envía a ese lugar, sino se elige otra máquina y la prueba.

Luego de "n" pruebas negativas el algoritmo termina y el proceso se ejecuta en la máquina de origen, por ser la que tiene la menor carga.

- **Un algoritmo de remates.** Utiliza un modelo económico con:
  - Compradores (procesos) y vendedores (procesadores) de servicios.
  - Precios establecidos por la oferta y la demanda.

Los procesos deben comprar tiempo de cpu. Cada procesador anuncia su precios, reúnen ofertas de los procesos, informa la ganadora, lo ejecuta y actualiza los precios.

10.15 PLANIFICACION EN SISTEMAS DISTRIBUIDOS: cada procesador hace su planificación local. La planificación independiente no es eficiente cuando se ejecutan en distintos procesadores un grupo de procesos relacionados entre sí, con gran atracción entre ellos.

Se debe garantizar que los procesos con comunicación frecuente se ejecuten de manera simultánea. Un grupo de procesos relacionados entre sí se iniciaran juntos, en muchos casos.

La comunicación dentro de los grupos debe prevalecer sobre la comunicación entre los grupos.

El algoritmo de "Austerhout" utiliza la planificación. Debe garantizar que todos los miembros del grupo se ejecuten al mismo tiempo. Cada procesador utiliza un algoritmo de planificación ROUND ROBIN. Se mantienen sincronizados los intervalos de tiempo. Todos los miembros del grupo se colocan en el mismo número de espacio de tiempo pero en procesadores distintos.