CAPITULO 24

THREADS

24.1. - **RPC CON HILOS**

La idea de usar hilos en RPC es aligerar a este último.

RPC local

Al iniciar un hilo servidor S, éste exporta su interfase informándosela al núcleo. La interfase define los procedimientos que puede llamar, sus parámetros, etc. Al iniciar un hilo cliente C este importa la interfase del núcleo y se le proporciona un identificador especial para utilizarlo en la llamada, entonces el núcleo sabe que C va a llamar a S, y crea estructuras de datos especiales con el fin de prepararse para la llamada. Una de estas estructuras es la pila compartida entre C y S y que se asocia de manera lectura/escritura en ambos espacios de direcciones.

Para llamar al servidor, C coloca sus argumentos en dicha pila utilizando para ello el procedimiento normal de transferencia y después hace una interrupción al núcleo al colocar un identificador especial en un registro. El núcleo se da cuenta de esto y sabe que es una llamada local, desarrollando las siguientes acciones: Modifica el mapa de memoria del cliente para colocarlo en el espacio de direcciones del servidor y dentro de este inicializa el hilo cliente al ejecutar el procedimiento del servidor. Esto se realiza de forma tal que los argumentos se encuentran ya en su lugar, o sea que no necesitan ser copiados.

Otra forma de agilizar el RPC local es haciendo desvanecer un hilo servidor (desaparece su pila e información de contexto) cuando termina de realizar una solicitud ya que es raro que deba tener variables locales y los datos en sus registros no tienen significado ya.

Recepción Implícita

Al llegar un nuevo mensaje a la máquina servidora, el núcleo crea en ese momento un nuevo hilo para darle servicio a la solicitud, además asocia el mensaje con el espacio de direcciones del servidor y configura la pila del nuevo hilo para tener acceso al mensaje (hilo de aparición instantánea - pop-up thread).

En este método los hilos no tienen que bloquearse en espera de más trabajo y por lo tanto no es necesario guardar información del contexto. Además la creación de un nuevo hilo es más económica que la restauración de uno ya existente y se ahorra tiempo al no tener que copiar los mensajes recibidos a un buffer dentro de un hilo servidor.

24.2. - **MODELOS DE SISTEMAS**

Los procesadores de un sistema distribuido se pueden organizar de distintas formas. A continuación veremos tres formas que se basan en diferentes filosofías respecto de lo que debe ser un sistema distribuido.

24.2.1. - El Modelo de Estación de Trabajo

Este modelo es directo, o sea el sistema consta de estaciones de trabajo, computadoras personales de alta calidad dispersas en un edificio y conectadas entre sí por medio de una LAN de alta velocidad. Las estaciones de trabajo pueden tener más de un usuario, pero solo uno a la vez puede estar conectado a ella o la estación puede estar inactiva.

Existen dos tipos de estaciones de trabajo, con disco y sin disco. Si la estación de trabajo no tiene disco el sistema de archivos debe manejarse por medio de uno o más servidores de archivos en la red.

Los usuarios tienen una cantidad fija de poder de cómputo exclusivo, garantizando un tiempo de respuesta. Cada usuario tiene un alto grado de autonomía y puede asignar los recursos de su estación de trabajo como juzgue sea necesario.

Una desventaja es que existen baches en que los usuarios no utilizan las máquinas y es probable que otros usuarios necesiten una capacidad de cómputo mayor y no puedan obtenerla. Una solución es usar estas estaciones de trabajo inactivas.

a) Si las estaciones de trabajo carecen de disco: el sistema de archivos debe ser implementado mediante uno o más servidores de red y las solicitudes de lectura/ escritura serán enviadas al servidor de archivos que realiza el trabajo y envía luego las respuestas. Tienen fácil mantenimiento y menor precio. Además, proporcionan simetría y flexibilidad.

b) Si las estaciones tienen sus propios discos: estos pueden ser usados de cuatro maneras:

i) Paginación y archivos temporales: Aunque es conveniente mantener los archivos de un usuario en servidores centrales de archivo para facilitar el respaldo y mantenimiento, también se necesitan discos para paginación y archivos temporales, si se usa el disco local se reduce el tráfico en la red.

ii) Paginación, archivos temporales y códigos objeto del sistema: Los discos locales también contienen programas ejecutables como ser compiladores, editores y manejadores de correo electrónico.

Al disponer una nueva versión del programa se transmite a todas las máquinas, pero si una máquina estaba apagada en ese instante queda con la versión vieja.

iii) Paginación, archivos temporales, códigos objeto del sistema y ocultamiento de archivos: Utiliza los discos locales como cache durante el acceso a un archivo del servidor, además de usarlo para paginación, archivos temporales y códigos binario. Así los usuarios pueden cargar archivos desde los servidores hasta sus propios discos, trabajarlos localmente y luego restaurarlos. Con esto se logra mantener centralizado el almacenamiento a largo plazo. Pero tiene un problema cuando dos usuarios utilizan el mismo archivo.

iv) Sistema local de archivo completo: Cada máquina puede tener su propio sistema de archivos autocontenido, pudiendo montar y tener acceso a los sistemas de archivos de otras máquinas. El objetivo de esto es que cada máquina esté autocontenida con un limitado contacto con el mundo exterior. Esto conlleva una pérdida de transparencia. El sistema resultante se parece más a un sistema operativo de red que a un sistema distribuido.

Uso del Disco	**Ventajas**	**Desventajas**
Sin disco	Bajo costo, fácil mantenimiento del hardware y el software, simetría y flexibilidad	Gran uso de la red, los servidores de archivos se pueden convertir en cuellos de botella
Paginación, archivos de tipo borrador	Reduce la carga de la red comparada con el caso sin disco	Un costo alto debido al gran número de discos necesarios
Paginación	Reduce todavía más la carga sobre la red	Alto costo, complejidad adicional para actualizar los códigos objeto
Paginación, archivos de tipo borrador, códigos objeto, ocultamiento de archivos	Una carga aún menor en la red, también reduce la carga en los servidores de archivos	Alto costo, problemas de consistencia del cache
Sistema local de archivos completo	Escasa carga en la red, elimina la necesidad de los servidores de archivos	Pérdida de transparencia

24.2.1.1. - Uso de estaciones de trabajo inactivas

Uno de los intentos de tratar de aprovechar la potencia de cómputo de una estación de trabajo inactiva consiste en el uso del Remote Login (RSH).

rsh máquina comando

Esto tiene varios problemas:

1) **Como localizar una estación de trabajo inactiva ?**

Una estación donde ningún usuario está conectado puede ejecutar decenas de procesos (demonios de correo, noticias, reloj, etc.), más aún, un usuario puede estar conectado pero si hace varios minutos que no toca el teclado o si la máquina no está ejecutando algún proceso en concreto se puede decir que dicha máquina se encuentra inactiva.

Los algoritmos para localizar estas estaciones se dividen en dos:

Controladas por servidor : cuando una estación de trabajo esta inactiva anuncia su disponibilidad nombre, dirección y propiedades en un archivo de registro en el servidor o coloca un mensaje que se anuncia en toda la red, registrando esto todas las otras estaciones. No obstante ello en este caso puede suceder que dos estaciones que buscan una inactiva detecten la misma y traten de iniciar un proceso al mismo tiempo.

Controladas por cliente : el cliente transmite una solicitud donde indica el programa que desea ejecutar y la cantidad de memoria necesaria, si necesita coprocesador, etc. (esto no es necesario si las estaciones son iguales). Todas las que están libres envían un mensaje a la que solicitó y ella decide con cuál trabajar. Si el mensaje de respuesta se demora directamente proporcional a la carga de la máquina que responde las respuestas de las máquinas con menor carga llegarán antes.

2) **Como lograr que un proceso remoto se ejecute en forma transparente ?**

Necesita la misma visión del sistema de archivos, el mismo directorio de trabajo, las mismas variables del ambiente si existen para poder ejecutar en la remota. Algunas de las llamadas al sistema se deben devolver a la máquina de origen sin hacer distinción alguna, otras nunca se pueden ejecutar en la máquina remota como ser lectura del teclado. Las llamadas al sistema relacionadas con el tiempo son un inconveniente puesto que los relojes de las distintas máquinas no tienen porqué estar sincronizados.

3) **Que ocurre si el usuario de la máquina inactiva regresa ?**

Lo más fácil es no hacer nada, pero esto va en contra de la idea de estaciones personales de trabajo. Otra posibilidad es eliminar el proceso intruso de manera abrupta y sin previo aviso (pérdida de trabajo y sistema de archivos caótico). Es mejor darle al proceso una advertencia y hacer esto con bondad. Otra solución es hacer que el proceso migre a otra máquina, ya sea a la máquina de origen o alguna otra inactiva. Esto último es posible pero tiene dificultades prácticas sustanciales, como por ejemplo lograr que migre el proceso padre y además todos sus hijos, debe llevarse sus buzones, conexiones de red, etc.

24.2.2. - El Modelo de la Pila de Procesadores

El problema que trata de resolver este esquema es cómo proveer a los usuarios de la potencia de cómputo de más de un CPU.

Lo que se hace es construir una pila de procesadores en el cuarto de máquinas los cuales se pueden asignar a los usuarios en forma dinámica. Los usuarios usan terminales gráficas de alto rendimiento, como las terminales X.

Desde el punto de vista conceptual se parecen mucho más al tiempo compartido tradicional que al modelo de computadora personal.

Si el sistema de archivos se debe concentrar en un pequeño número de servidores de archivos, debe ser posible hacer lo mismo con los servidores de cómputo. De hecho se convierte todo el poder de cómputos en " Estaciones de trabajo inactivas " a las que se puede tener acceso en forma dinámica. Este modelo proviene de las teorías de colas.

El estudio nos dice que si reemplazamos N pequeños recursos por uno grande que sea N veces mayor podemos reducir el tiempo promedio de respuesta N veces. Este resultado de la teoría de colas es uno de los principales argumentos en contra de los sistemas distribuidos. Los S.D. tienen a favor el costo que será mucho menor, también la confiabilidad y la tolerancia a fallas. Además de esto las estaciones de trabajo tienen una respuesta uniforme independientemente de lo que hagan las demás personas.

Además esta prestación del modelo de pila de procesadores se basa en que una solicitud se puede dividir para ejecutarse en los N procesadores de la pila si la solicitud solo puede partirse entre 3 procesadores el desempeño solo mejorará en un tercio.

No obstante, este modelo es mejor que el trabajo de buscar estaciones inactivas.

Por último la elección depende en gran medida del tipo de tareas que se realicen. Si los usuarios trabajan con edición de archivos y envían cada tanto algún mensaje de correo, probablemente baste con el modelo de estaciones de trabajo, en tanto que si un grupo de usuarios trabaja en el desarrollo de un gran proyecto de software, o hacen grandes simulaciones, probablemente será mejor el esquema de pila de procesadores.

24.2.3. - Un Modelo Híbrido

Se puede tener una estación de trabajo personal para cada usuario y además tener una pila de procesadores dedicada a los procesos no interactivos y a todo el cómputo pesado. No se realiza el trabajo de búsqueda de estaciones inactivas lo que hace más sencillo el diseño del sistema.

Si bien este esquema es el más caro combina las ventajas de los dos anteriores.

24.3. - **COMO SE ASIGNAN LOS PROCESADORES?**

El problema a tratar aquí es cómo asignar un procesador a un proceso. Por ejemplo en el modelo de estaciones de trabajo cuando asignar el procesador local o cuándo buscar una máquina inactiva; en el modelo de pila cómo asignar los procesadores cuando aparece un nuevo proceso.

Las estrategias de asignación de procesadores se pueden dividir en dos categorías :

No migratoria: Al crearse un proceso se decide donde ponerlo y permanece allí hasta que termina.

Migratorios: Un proceso puede trasladarse aunque haya iniciado su ejecución.

Evidentemente la idea subyacente a la asignación de procesadores es la optimización de algún componente del sistema. Sin embargo lo que se desee optimizar variará de un sistema a otro.

Una optimización posible es el **uso de la CPU**, es decir evitar a toda costa el tiempo ocioso de la misma.

Otro parámetro a tener en cuenta es **el tiempo promedio de respuesta** o su variante, **la tasa de respuesta** que se define como la cantidad de tiempo necesaria para ejecutar un proceso en cierta máquina dividido por el tiempo que tardaría en ejecutarse en un procesador de referencia no cargado.

Aspectos de diseño

Para diseñar estos algoritmos los diseñadores deben tomar decisiones con respecto a:

1) Algoritmos Deterministas vs. Heurísticos: Cuando se tiene de antemano toda la información sobre el comportamiento de los procesos se utilizan algoritmos determinísticos en tanto que los heurísticos se usan cuando la carga es impredecible.

2) Algoritmos Centralizados vs. Distribuidos: Se han propuesto algoritmo centralizados por la carencia muchas veces de buenas alternativas distribuidas.

3) Algoritmos óptimos vs. Subóptimos: Obtener el óptimo implica mayor complejidad de cálculo. La mayoría de los sistemas distribuidos reales buscan soluciones subóptimas, heurísticas y distribuidas debido a la dificultad para obtener las óptimas.
4) Algoritmos Locales vs. Globales: Para estos algoritmos se debe tener en cuenta que al crearse un proceso si hay que tomar la decisión de ejecutarlo en otra máquina (**política de transferencia**) la opción consiste en basar o no esta decisión por completo en la información local. Los algoritmos locales son sencillos pero están muy lejos de ser los óptimos, mientras que los globales sólo dan un resultado un poco mejor a un mayor costo.
5) Algoritmos Iniciados por el Receptor vs Iniciados por el Emisor: Una vez que se decidió liberarse de un proceso la **política de localización** debe decidir dónde enviarlo. Esta política no puede ser local, necesita información de la carga en todo el sistema. Esta información se puede dispersar como inicio de intercambio por parte de los emisores o por parte de los receptores.

24.4. - **PLANIFICACIÓN EN SISTEMAS DISTRIBUIDOS**

Por lo general cada procesador hace su propia planificación local, sin embargo, si un grupo de procesos relacionados entre sí y con una gran interacción, se ejecutan en distintos procesadores la planificación independiente no es eficiente.

Lo que se necesita es una forma de garantizar que los procesos con comunicación frecuente se ejecuten en forma simultánea.

Aunque es difícil determinar en forma dinámica los patrones de comunicación entre los procesos, en muchos casos, un grupo de procesos relacionados entre sí se iniciarán juntos. Supongamos que los procesos se crean en grupos y que la comunicación dentro de los grupos prevalece sobre la comunicación entre los grupos, y que se tiene un número de procesadores lo bastante grande como para manejar al grupo de mayor tamaño y que cada procesador se multiprograma con N espacios para los procesos.

Existe un concepto llamado COPLANIFICACION (Ousterhout 1982) el cual toma en cuenta los patrones de comunicación entre los procesos durante la planificación para garantizar que todos los miembros de un grupo se ejecuten al mismo tiempo.

Existen algoritmos para planificar esto:

Uno se realiza mediante una matriz con N entradas para los procesadores (columnas) y M para los espacios de tiempo. Así la columna 4 tiene todos los procesos que se ejecutan en el procesador 4 a través de los tiempos, el renglón 3 es la colección de todos los procesos en el espacio 3 para algún procesador.

Los procesadores ejecutan el proceso del espacio 0 durante un cierto tiempo luego todos los del 1 y así siguiendo usando un algoritmo round-robin. Se puede utilizar un mensaje para indicarle a cada procesador el momento en que debe intercambiar los procesos y mantener sincronizados los intervalos de tiempo.

Si todos los miembros de un grupo se colocan en el mismo número de espacio pero en procesadores distintos se obtiene paralelismo de nivel N, ejecutándose todos al mismo tiempo y maximizando el desempeño de la comunicación.

Una variante para mejorar sería separar la matriz por renglones y concatenarlos formando un gran renglón. Con k procesadores, k entradas cualquiera consecutivas pertenecen a distintos procesadores. Para asignar un nuevo grupo de procesos a las entradas se deja una ventana de k entradas de ancho en el renglón de gran tamaño de modo que la entrada del extremo izquierdo esté vacía, pero que la entrada justo a la izquierda de la ventana esté ocupada. Si existe el número suficiente de entradas en dicha ventana, los procesos se asignan a las entradas vacías o bien la ventana se desliza hacia la derecha y se repite el algoritmo.

La planificación se lleva a cabo, al iniciar la ventana en la orilla izquierda y moviéndola a la derecha, tantas entradas como tenga la ventana por cada intervalo de tiempo, teniendo cuidado de no dividir los grupos en las ventanas.