Estructura o Arquitectura del sistema y ejemplos de distintos SO. Stalling: pp. 76-101

Página 76

Debido al tamaño de los sistemas operativos grandes ha habido un incremento en el uso de los conceptos de capas jerárquicas y abstracción de información. La estructura jerárquica de un SO separa sus funciones de acuerdo a las características de su escala de tiempo y su nivel de abstracción. Se puede ver el SO como una serie de niveles, de forma que cada nivel realiza un subconjunto relacionado de funciones requeridas por el SO. Dicho nivel confía en los niveles inmediatamente inferiores para realizar funciones más primitivas y ocultar los detalles de esas funciones. Cada nivel proporciona servicios a la capa inmediatamente superior. Idealmente, estos niveles han de definirse de forma que los cambios en un nivel no requiera cambios en otros niveles. De esta forma, descomponemos un problema complejo (el SO), en una serie de problemas más sencillos.

Notar que las capas inferiores, en general, tratan con una escala de tiempo menor.

Página 79

Debido a la evolución que han ido sufriendo los SOs ha ido surgiendo la necesidad de desarrollar nuevas formas de estructurarlos, en general, todo el trabajo realizado se puede clasificar en uno de los 3 grupos:

- Arquitectura micronúcleo o microkernel
- Multihilo
- Multiprocesamiento simétrico
- Sistemas operativos distribuidos
- Diseño orientado a objetos

Página 80

Hasta hace relativamente poco tiempo, la mayoría de los sistemas operativos estaban formados por un gran **núcleo monolítico**. Estos grandes núcleos proporcionan la mayoría de las funcionalidades consideradas propias del sistema operativo, incluyendo la planificación, los sistemas de ficheros, las redes, los controladores de dispositivos, la gestión de memoria y otras funciones. Normalmente, un núcleo monolítico se implementa como un único proceso, con todos los elementos compartiendo el mismo espacio de direcciones.

Una **arquitectura micronúcleo** asigna sólo unas pocas funciones esenciales al núcleo, incluyendo los espacios de almacenamiento, comunicación entre procesos (IPC), y la planificación básica. Ciertos procesos proporcionan otros servicios del sistema operativo, algunas veces denominados servidores, que ejecutan en modo usuario y son tratados como cualquier otra aplicación por el micronúcleo. Esta técnica desacopla el núcleo y el desarrollo del servidor. La técnica micronúcleo simplifica la implementación, proporciona flexibilidad y se adapta perfectamente a un entorno distribuido. En esencia, un micronúcleo interactúa con procesos locales y remotos del servidor de la misma forma, facilitando la construcción de los sistemas distribuidos.

Multithreading es una técnica en la cual un proceso, ejecutando una aplicación, se divide en una serie de hilos o *threads* que pueden ejecutar concurrentemente. Se pueden hacer las siguientes distinciones:

- ***Thread* o hilo.** Se trata de una unidad de trabajo. Incluye el contexto del procesador (que contiene el contador del programa y el puntero de pila) y su propia área de datos para una pila (para posibilitar el salto a subrutinas). Un hilo se ejecuta secuencialmente y se puede interrumpir de forma que el procesador pueda dar paso a otro hilo.

- **Proceso.** Es una colección de uno o más hilos y sus recursos de sistema asociados (como la memoria, conteniendo tanto código, como datos, ficheros abiertos y dispositivos). Esto corresponde al concepto de programa en ejecución. Dividiendo una sola aplicación en múltiples hilos, el programador tiene gran control sobre la modularidad de las aplicaciones y la temporización de los eventos relacionados con la aplicación.

El multithreading es útil para aplicaciones que llevan a cabo un número de tareas esencialmente independientes que no necesitan ser serializadas.

Para lograr mayor eficiencia y fiabilidad, una técnica consiste en emplear multiprocesamiento simétrico (SMP: Symmetric Multi- Processing), un término que se refiere a la arquitectura hardware del computador y también al comportamiento del sistema operativo que explota dicha arquitectura. Se puede definir un multiprocesador simétrico como un sistema de computación aislado con las siguientes características:

1. Tiene múltiples procesadores.

2. Estos procesadores comparten las mismas utilidades de memoria principal y de E/S, interconectadas por un bus de comunicación u otro esquema de conexión interna.

3. Todos los procesadores pueden realizar las mismas funciones (de ahí el término simétrico).

Página 81

El SO de un SMP planifica procesos o hilos a través de todos los procesadores. Las ventajas de SMP sobre una arquitectura monoprocesador son las siguientes:

- Rendimiento. Se pueden ejecutar múltiples procesos simultáneamente.
- Disponibilidad. El fallo de un solo procesador no para la máquina.
- Crecimiento incremental. Se puede mejorar el sistema añadiendo un procesador adicional.
- Escalado. Los fabricantes pueden ofrecer un rango de productos con diferente precio y características basadas en el número de procesadores configurado en el sistema.

Estas ventajas son beneficios potenciales, no garantizados, debido a que se necesita que el SO sea capaz de explotar el paralelismo de un sistema SMP.

Página 82

Multithreading y SMP se suelen analizar juntos, pero tanto el primero es útil en sistemas monoprocesador, debido a que puede ayudar para estructurar aplicaciones y procesos del núcleo, como el segundo es útil en sistemas sin hilos, debido a que los procesos se podrán ejecutar en paralelo.

Una característica interesante de SMP es que la existencia de múltiples procesadores es totalmente transparente al usuario puesto que el SO se encarga de todo. Un tema diferente es proporcionar la apariencia de un solo sistema a un cluster de computadores separado. En este caso se trata de una

colección de entidades cada uno con sus propios módulos de memoria principal, de memoria secundaria y de otros módulos de E/S.

Un sistema operativo distribuido proporciona la ilusión de un solo espacio de memoria principal y un solo espacio de memoria secundario. Pese a que los clusters se están volviendo cada día más populares, el estado del arte de los sistemas operativos distribuidos está retrasado con respecto a los monoprocesadores y los SOs SMP.

Otra innovación es el diseño de los SOs es el uso de tecnologías orientadas a objetos. Este tipo de diseño introduce una disciplina al proceso de añadir extensiones modulares a un pequeño núcleo. A nivel del SO, una estructura basada en objetos permite al programador personalizar un SO sin eliminar la integridad del sistema. La orientación a objetos también facilita el desarrollo de herramientas distribuidas y SOs distribuidos.

NOTA: A partir de la descripción global de Microsoft Windows hasta la página 101 no hemos apuntado nada, si sobre tiempo, leerlo como curiosidad y por si se atreviese a poner algo la profe.

Micronúcleos (o microkernels). Stalling pp. 176-181

Página 176

Un micronúcleo es la pequeña parte central de un SO que proporciona las bases para extensiones modulares.

ARQUITECTURA MICRONÚCLEO
Como vimos previamente, los primeros SOs eran monolíticos, de prácticamente cualquier procedimiento se podía llamar a cualquier otro. Esto se hizo insostenible a medida que los SOs fueron creciendo, y entonces se empezó a adoptar la estructura por capas que vimos previamente. Pero en este enfoque, sigue habiendo muchos problemas, cada capa suele tener muchas funcionalidades, de forma que cambios en una capa pueden tener numerosos efectos, los cuales pueden ser difíciles de seguir en el código, en las capas adyacentes (por encima o por debajo). Además, es difícil construir la seguridad puesto que hay muchas interacciones entre capas adyacentes.

La filosofía del micronúcleo es que únicamente las funciones esenciales del SO estén en el núcleo.

La arquitectura del micronúcleo reemplaza la tradicional estructura vertical y estratificada en capas por una horizontal. Los componentes del SO externos al núcleo se implementan como servidores de procesos, de forma que interactúan entre ellos dos a dos, normalmente por paso de mensajes a través del micronúcleo.

De esta forma, el micronúcleo funciona como intercambiador de mensajes y además, realiza una función de protección, puesto que previene el paso de mensajes a no ser que el intercambio esté permitido.

Por ejemplo, si una aplicación quiere abrir un archivo, manda un mensaje al servidor del sistema de archivos. Si quiere crear un proceso o hilo, manda un mensaje al servidor de procesos. Es decir, tenemos una arquitectura cliente/servidor dentro del ordenador.

Página 177

BENEFICIOS DE UNA ORGANIZACIÓN MICRONÚCLEO
Los beneficios de una organización micronúcleo son:

- Interfaces uniformes
- Extensibilidad
- Flexibilidad
- Portabilidad
- Fiabilidad
- Soporte de sistemas distribuidos
- Soporte de SOs orientados a objetos

Página 178

El micronúcleo impone una interfaz uniforme en las peticiones realizadas por un proceso. Los procesos no necesitan diferenciar entre servicios a nivel de núcleo y a nivel de usuario puesto que todos los servicios se proporcionan a través de paso de mensajes.

La arquitectura micronúcleo facilita la extensibilidad, permitiendo agregar nuevos servicios. Además, cuando se añade una nueva característica, solo el servidor relacionado necesita modificarse o añadirse.

Respecto a la flexibilidad, este tipo de arquitectura, además de permitir añadir nuevas características al SO como hemos visto, también permite la eliminación de estas, de forma que tendríamos una implementación más pequeña y eficiente ya que no todo el mundo puede necesitar todas las características

Respecto a la portabilidad, debido a que todo o gran parte del código específico del procesador se encuentra en el micronúcleo, por lo que los cambios necesarios para transferir el sistema a un nuevo procesador son menores y tienden a estar unidos en grupos lógicos.

Un micronúcleo pequeño se puede verificar de forma rigurosa, aumentando así la fiabilidad del mismo. Ya sabemos que cuanto mayor sea el sistema, más difícil de asegurar que sea fiable es.

Respecto al soporte para sistemas distribuidos (clusters) tenemos que tener en cuenta que cuando se envía un mensaje desde un cliente hasta un proceso servidor, el mensaje debe incluir el identificador del servicio pedido, pues bien si se configura un sistema distribuido de tal forma que todos los procesos y servicios tengan un identificador único, entonces habrá una sola imagen del sistema a nivel de micronúcleo, un proceso podrá enviar un mensaje sin saber en qué máquina está.

Una arquitectura micronúcleo funciona bien en el contexto de un SO orientado a objetos. Un enfoque orientado a objetos puede servir para diseñar el micronúcleo y para desarrollar extensiones modulares para el SO. Un enfoque prometedor para casar la arquitectura micronúcleo con los principios de OOOS es el uso de componentes. Los componentes son objetos con interfaces claramente definidas que pueden ser interconectadas para la realización de software a través de bloques de construcción. Toda la interacción entre componentes utiliza la interfaz del componente.

RENDIMIENTO DEL MICRONÚCLEO

Una potencial desventaja que se cita a menudo de los micronúcleos es el rendimiento. Lleva más tiempo construir y enviar un mensaje a través del micronúcleo, y aceptar y decodificar la respuesta, que hacer una simple llamada a un servicio. Sin embargo también son importantes otros factores, de forma que es difícil generalizar sobre la desventaja de rendimiento, si es que la hay.

Una posible solución a esto puede ser hacer el micronúcleo más grande, volviendo a introducir servicios críticos y manejadores en el sistema operativo. De esta forma, al incrementar la funcionalidad del micronúcleo, se reduce el número de cambios de contexto usuario-núcleo y el número de cambios de espacio de direcciones de proceso. Sin embargo, esta solución, pese a que reduce los problemas de rendimiento, sacrifica la fortaleza del diseño del micronúcleo (mínimas interfaces, flexibilidad…).

Otro enfoque consiste en hacer el micronúcleo aún más pequeño. Un micronúcleo muy pequeño apropiadamente diseñado elimina las pérdidas de rendimiento y mejora la flexibilidad y la fiabilidad.

DISEÑO DEL MICRONÚCLEO

Debido a que los diferentes micronúcleos presentan una gran variedad de tamaños y funcionalidades, no se pueden facilitar las reglas concernientes a qué funcionalidades deben darse por parte del micronúcleo y qué estructura debe implementarse. Pese a esto, este es el conjunto mínimo de funciones y servicios que debería tener implementado.

El micronúcleo debe incluir aquellas funciones que dependen directamente del hardware y aquellas funciones necesarias para mantener a los servidores y aplicaciones operando en modo usuario. Estas funciones están dentro de las categorías generales de gestión de memoria a bajo nivel, intercomunicación de procesos (IPC), y E/S y manejo de interrupciones.

Gestión de memoria a bajo nivel.

El micronúcleo tiene que controlar el concepto hardware de espacio de direcciones para hacer posible la implementación de protección a nivel de proceso. Con tal de que el micronúcleo se responsabilice de la asignación de cada página virtual a un marco físico, la parte principal de gestión de memoria, incluyendo la protección del espacio de memoria entre procesos, el algoritmo de reemplazo de páginas y otra lógica de paginación, pueden implementarse fuera del núcleo. Por ejemplo, un módulo de memoria virtual fuera del micronúcleo decide cuándo traer una página a memoria y qué página presente en memoria debe reemplazarse; el micronúcleo proyecta estas referencias de página en direcciones físicas de memoria principal.

Se recomienda un conjunto de tres operaciones de micronúcleo que pueden dar soporte a la paginación externa y a la gestión de memoria virtual:

- Conceder (Grant). El propietario de un espacio de direcciones (un proceso) puede conceder alguna de sus páginas a otro proceso. El núcleo borra estas páginas del espacio de memoria del otorgante y se las asigna al proceso especificado.

- Proyectar (Map). Un proceso puede proyectar cualquiera de sus páginas en el espacio de direcciones de otro proceso, de forma que ambos procesos tienen acceso a las páginas. Esto

genera memoria compartida entre dos procesos. El núcleo mantiene la asignación de estas páginas al propietario inicial, pero proporciona una asociación que permite el acceso de otros procesos.

- Limpiar (Flush). Un proceso puede reclamar cualquier página que fue concedida o asociada a otro proceso.

Para comenzar, el núcleo asigna toda la memoria física disponible como recursos a un proceso base del sistema. A medida que se crean los nuevos procesos, se pueden conceder o asociar algunas páginas del espacio de direcciones original a los nuevos procesos. Este esquema podría dar soporte a múltiples esquemas de memoria virtual simultáneamente.

Página 180

Comunicación entre procesos (IPC)

La forma básica de comunicación entre dos procesos o hilos en un SO con micronúcleo son los mensajes.

Un mensaje incluye una cabecera que identifica a los procesos remitente y receptor y un cuerpo que contiene directamente los datos, un puntero a un bloque de datos, o alguna información de control del proceso. Normalmente podemos pensar que las IPC se fundamentan en puertos asociados a cada proceso. Un puerto es, en esencia, una cola de mensajes destinada a un proceso particular; un proceso puede tener múltiples puertos. Asociada a cada puerto existe una lista que indica qué procesos se pueden comunicar con éste. Las identidades y funcionalidades de cada puerto se mantienen en el núcleo. Un proceso puede concederse nuevas funcionalidades mandando un mensaje al núcleo con las nuevas funcionalidades del puerto.

Página 181

Gestión de E/S e interrupciones

Con una arquitectura micronúcleo es posible manejar las interrupciones hardware como mensajes e incluir los puertos de E/S en los espacios de direcciones. El micronúcleo puede reconocer las interrupciones pero no las puede manejar. Más bien, genera un mensaje para el proceso a nivel de usuario que está actualmente asociado con esa interrupción. De esta forma, cuando se habilita una interrupción, se asigna un proceso de nivel de usuario a esa interrupción y el núcleo mantiene las asociaciones. La transformación de las interrupciones en mensajes las debe realizar el micronúcleo, pero el micronúcleo no está relacionado con el manejo de interrupciones específico de los dispositivos.

Se sugiere ver el hardware como un conjunto de hilos que tienen identificadores de hilo único y que mandan mensajes (únicamente con el identificador de hilo) a hilos asociados en el espacio de usuario. El hilo receptor determina si el mensaje proviene de una interrupción y determina la interrupción específica.

Estructura de los SO. Carretero pp. 66-69

Página 66

ESTRUCTURA DEL SO
Los SOs se pueden clasificar de acuerdo a su estructura, en SOs monolíticos o estructurados.

Un sistema operativo monolítico no tiene una estructura bien clara y definida. Todos sus componentes se encuentran integrados en un único programa (el SO) que ejecuta en un único espacio de direcciones. Además, todas las funciones que ofrece se ejecutan en modo privilegiado. Ejemplos claros de este tipo de sistemas son MS-DOS o UNIX. El problema principal viene cuando queremos modificar o ampliar el SO, o cuando vemos que no se sigue el principio de ocultación de la información.

Hablando de SOs estructurados diferenciamos los sistemas por capas y los sistemas cliente-servidor.

La estructura por capas ya la sabemos de otros apuntes. Y ya sabemos que su principal ventaja es la modularidad y la ocultación de la información. Un ejemplo es el OS/2, descendiente de MS-DOS.

El modelo cliente-servidor es el del micronúcleo, ya lo hemos estudiado en otros apuntes también. El SO está formado por diversos módulos y cada uno de estos se puede desarrollar por separado.

Página 67

El modelo de máquina virtual se basa en un monitor capaz de suministrar m versiones del hardware, es decir, m máquinas virtuales. Cada copia es una réplica exacta del hardware, incluso con modo privilegiado y modo usuario, por lo que sobre cada una de ellas se puede instalar un SO convencional. Una petición de servicio es atendida por la copia de SO sobre la que ejecuta. Cuando dicho SO desea acceder al hardware, por ejemplo, para leer de un periférico, se comunica con su MV, como si de una máquina real se tratase. Sin embargo, con quien se está comunicando es con el monitor de máquina virtual que es el único que accede a la máquina real.

Otra forma de construir una máquina virtual es situando el monitor de MV sobre otro SO, no directamente sobre el hardware.

Las ventajas e inconvenientes de las MV son:

- Añade un nivel de multiprogamación, puesto que cada máquina virtual ejecuta sus propios procesos.
- Añade un nivel adicional de aislamiento, de forma que si se compromete la seguridad en un SO no se compromete en los demás. Esta funcionalidad puede ser muy importante cuando se están haciendo pruebas.
- Si lo que se genera es una máquina estándar, se consigue tener una plataforma de ejecución independiente del hardware, por lo que no hay que recompilar los programas para pasar de un computador a otro.
- Tiene el inconveniente de añadir una sobrecarga computacional, lo que puede hacer más lenta la ejecución.

En algunos casos se incluye una capa, llamada exokernel, que ejecuta en modo privilegiado y que se encarga de asignar recursos a las máquinas virtuales y de garantizar que ninguna máquina utilice recursos de otra.

Algunos fabricantes de hardware están incluyendo modos especiales de ejecución para facilitar la fabricación de las MV.

Página 69

Un SO distribuido es un SO diseñado para gestionar un multicomputador. El usuario percibe un único SO centralizado, haciendo, por tanto, más fácil el uso de la máquina. Un SO de este tipo sigue teniendo las mismas características que un SO convencional pero aplicadas a un sistema distribuido. Estos sistemas no han tenido mucho éxito comercial y se han quedado en la fase experimental.

Un middleware es una capa de software que se ejecuta sobre un SO ya existente y que se encarga de gestionar un sistema distribuido o un multicomputador. En este sentido, presenta una funcionalidad similar a la de un SO distribuido. La diferencia es que ejecuta sobre SOs ya existentes que puedan ser, además, distintos, lo que hace más atractiva su utilización. Ejemplos de middleware: DCOM, DCE…

Estructura de los SO. Silberschatz pp. 52-62

Página 53

ESTRUCTURA SIMPLE
Muchos sistemas no tienen una estructura bien definida. Frecuentemente, estos sistemas comienzan siendo pequeños, simples y limitados y luego van creciendo. Un ejemplo de esto es MS-DOS o UNIX. En UNIX, todo lo que está por debajo de la interfaz de llamadas al sistema y por encima del hardware es el kernel, el cual proporciona una enorme cantidad de funciones en un sólo nivel. Esta estructura monolítica es difícil de implementar y mantener.

ESTRUCTURA EN NIVELES
Con el soporte hardware apropiado, los SOs pueden dividirse en partes más pequeñas y adecuadas que lo que permitían los SOs originales MS-DOS y UNIX, de forma que el SO puede mantener un control mucho mayor sobre la computadora y sobre las aplicaciones que hacen uso de esta.

La estructura en niveles es una de la posibles formas de hacer un sistema modular, el SO se divide en una serie de capas. De forma que la capa 0 es el hardware y la capa más alta es la interfaz de usuario. De esta forma, si tenemos una capa M, todas las capas por encima podrán invocar las rutinas y las estructuras de datos de las que disponga esta capa. De igual forma, la capa M, podrá invocar las de las capas inferiores.

La principal ventaja de este método es la simplicidad de su construcción y de su depuración. El hecho de que cada nivel emplee funciones y servicios de los niveles inferiores facilita mucho la depuración del SO puesto que podemos empezar a depurar desde abajo, y cuando encontremos un fallo, sabremos que pertenece al nivel que estamos depurando, puesto que los niveles inferiores estarán ya checkeados.

Cada nivel se implementa utilizando solamente las operaciones proporcionadas por los niveles inferiores, de forma que este no necesitará saber como se implementan estas operaciones, solamente qué es lo que hacen. De esta forma se le oculta mucha información a los niveles superiores.

La principal dificultad de construir un SO de esta forma es definir correctamente los diferentes niveles.

Un problema que tienen este tipo de SOs es que suelen ser menos eficientes puesto que cuando se realiza una llamada al sistema, esta tendrá que ir viajando por todos los niveles hasta el hardware. De forma que cada nivel añade una carga adicional a cada llamada al sistema.

En los diseños más recientes se han empezado a construir SOs con menos niveles, ampliando la funcionalidad de cada uno de estos. De esta forma, se tienen muchas de las ventajas del código modular, a la vez que se evitan los problemas más difíciles relacionados con la definición e interacción de los niveles.

Página 55

MICROKERNELS

Este método elimina todos los componentes no esenciales del kernel y los implementa como programas del sistema y de nivel de usuario, de esta forma tenemos un kernel más pequeño. No hay consenso respecto a qué servicios deberían de permanecer en el kernel y cuales implementarse en el espacio de usuario.

Sin embargo, normalmente los microkernels proporcionan una gestión de memoria y de los procesos mínima, además de un mecanismo de comunicaciones.

La función principal del microkernel es proporcionar un mecanismo de comunicaciones entre el programa cliente y los distintos servicios que se ejecutan también en el espacio de usuario.

El hecho de que la mayoría de servicios se ejecuten como procesos de usuario y no como procesos del kernel otorga una mayor seguridad al sistema. Además, si uno de estos servicios falla, el resto del SO no se ve afectado.

El problema como ya hemos visto antes es que los microkernels pueden tener un rendimiento peor que otras soluciones debido a la carga de procesamiento adicional impuesta por las funciones del sistema. Un ejemplo de esto es Windows NT, la primera versión era microkernel con niveles, sin embargo, esta versión proporcionaba un rendimiento muy bajo, así que empezaron a pasar diversos niveles del espacio de usuario al espacio de kernel. Esto produjo que cuando se diseñó Windows XP, la arquitectura del SO era más monolítica que microkernel.

Página 56

MÓDULOS

Quizá la mejor metodología actual para diseñar SOs es la que usa técnicas de programación orientada a objetos para crear un kernel modular. En este caso, el kernel dispone de un conjunto de componentes fundamentales y enlaza dinámicamente los servicios adicionales, bien durante el arranque o en tiempo de ejecución.

Un diseño así permite al kernel proporcionar servicios básicos y también permite implementar ciertas características dinámicamente. Por ejemplo, se pueden añadir al kernel controladores de bus y de dispositivos para hardware específico y puede agregarse como módulo cargable el soporte para diferentes sistemas de archivo.

El resultado final es similar a un sistema de niveles, en el sentido de que cada sección del kernel

tiene interfaces bien definidas y protegidas, pero es más flexible que un sistema de niveles, porque cualquier módulo puede llamar a cualquier módulo. Además, el método también es similar a utilizar un microkernel, ya que el módulo principal sólo dispone de las funciones esenciales y de los conocimientos sobre cómo cargar y comunicarse con otros módulos; sin embargo, es más eficiente que el microkernel puesto que los módulos no necesitan invocar un mecanismo de paso de mensajes para comunicarse entre ellos.

Página 58

MÁQUINAS VIRTUALES

La idea fundamental que subyace a una máquina virtual es la de abstraer el hardware de la computadora, formando varios entornos de ejecución diferentes, creando así la ilusión de que cada entorno de ejecución está operando con su propia computadora privada.

Con los mecanismos de planificación de la CPU y las técnicas de memoria virtual, un SO puede crear la ilusión de que un proceso tiene su propio procesador con su propia memoria (virtual).

El método de máquina virtual proporciona una interfaz que es idéntica al hardware básico subyacente. Cada proceso dispone de una copia (virtual) de la computadora subyacente.

La razón principal de usar una MV es el poder compartir el mismo hardware y, a pesar de ello, operar con entornos de ejecución diferentes (es decir, diferentes SOs) de forma concurrente.

Una de las principales dificultades del método de máquina virtual son los sistemas de disco. Supongamos que la máquina física tiene 3 unidades de disco, pero desea dar soporte a siete máquinas virtuales. Claramente, no se puede asignar una unidad de disco a cada máquina virtual, dado que el propio software de la máquina virtual necesitará un importante espacio en disco para proporcionar la memoria virtual y los mecanismos de gestión de colas. La solución consiste en proporcionar discos virtuales, denominados minidiscos en el sistema operativo VM de IBM, que son idénticos en todo excepto en el tamaño.

El sistema implementa cada minidisco asignándole tantas pistas de los discos físicos como necesite el minidisco. Obviamente, la suma de todos los minidiscos tiene que ser menor que el espacio de disco físicamente disponible.

Esta arquitectura proporciona una forma muy útil de dividir el problema de diseño de un sistema interactivo multiusuario en dos partes más pequeñas.

Página 59

IMPLEMENTACIÓN

Aunque este concepto es muy útil, resulta difícil de implementar. Hay que tener en cuenta que en la MV debemos de simular un modo kernel (virtual) y un modo usuario (virtual). Esto se consigue de esta forma:

Cuando se hace una llamada al sistema por parte de un programa que se esté ejecutando en una MV en modo usuario virtual, se produce una transferencia al monitor de la MV en la máquina real.

Cuando el monitor de la MV obtiene el control, puede cambiar el contenido de los registros y del contador de programa para que la MV simule el efecto de la llamada al sistema. A continuación, puede reiniciar la MV, que ahora se encontrará en modo kernel virtual. La principal diferencia es el tiempo. Mientras que la E/S real puede tardar 100 msg, la E/S virtual puede llevar menos tiempo (puesto que se pone en cola) o más tiempo (puesto que es interpretada).

BENEFICIOS
En este tipo de entorno, existe una protección completa de los recursos del sistema puesto que cada MV está completamente aislada de las demás. Pese a esto, no es posible la compartición directa de recursos.

Para intentar permitir esta compartición se han implementado 2 métodos:

1. Compartir un minidisco y por tanto, compartir archivos.
2. Definir una red de comunicaciones virtual.

Aparte, el uso de máquinas virtuales es perfecto a la hora de investigar y desarrollar otros SOs, puesto que tienes tu sistema aislado y no se corre el peligro de tocar algo que no debes.

Página 60

Dos máquinas virtuales populares actualmente son VMware y la máquina virtual de Java. Estas MVs operan por encima de un sistema operativo de cualquier tipo, por lo que, los distintos métodos de diseño de SOs no son mutuamente excluyentes.

Sistemas Operativos de tiempo real: Stalling pp. 463-466

Página 463

El SO, y en particular, el planificador, es quizás el componente más importante de un sistema de tiempo real. Algunos ejemplos de sistemas de tiempo real pueden ser control de experimentos de laboratorio, robótica, control del tráfico aéreo, sistemas militares de mando y control…

La computación en tiempo real puede definirse como aquella en la que la corrección del sistema depende no sólo del resultado lógico de la computación sino también del momento en el que se producen los resultados.

En un sistema de tiempo real, algunas de las tareas son de tiempo real y estas tienen cierto grado de urgencia. Tales tareas intentan controlar o reaccionar a eventos que tienen lugar en el mundo real.

Estas tareas puedes ser clasificadas en duras y blandas. Una tarea de tiempo real duro es aquella que debe cumplir su plazo límite, de otro modo, se producirá un daño inaceptable o error fatal en el sistema. Una tarea de tiempo real suave/blanda tiene asociado un plazo límite deseable pero no obligatorio, sigue teniendo sentido planificar y completar la tarea incluso cuando se ha cumplido su plazo límite.

Estas tareas también puedes ser periódicas o aperiódicas. Las aperiódicas tienen un plazo en el cual debe finalizar o comenzar, o puede tener una restricción tanto de su instante de comienzo como de

finalización. En el caso de las tareas periódicas, el requisito puede ser enunciado como “una vez por periodo T” o “exactamente T unidades a parte”.

CARACTERÍSTICAS DE LOS SISTEMAS OPERATIVOS DE TIEMPO REAL

- Determinismo
- Reactividad
- Control de usuario
- Fiabilidad
- Operación de fallo suave

Un SO es determinista en el sentido de que realiza operaciones en instantes de tiempo fijos predeterminados o dentro de intervalos de tiempo predeterminados. Cuando múltiples procesos compiten por recursos y tiempo de procesador, ningún sistema puede ser totalmente determinista.
En un SO de tiempo real, las solicitudes de servicio de los procesos son dirigidas por eventos externos y temporizaciones. El grado en el que un SO puede satisfacer de manera determinista las solicitudes depende, primero, de la velocidad a la que es capaz de responder a las interrupciones, y, segundo, de si el sistema tiene capacidad suficiente para manejar todas las solicitudes dentro del tiempo requerido.

Una característica relacionada es la reactividad. Mientras que el determinismo se encarga de cuanto tiempo tarda el sistema en reconocer la interrupción, la reactividad se preocupa de cuánto tiempo tarda el SO, después del reconocimiento, en servirla. Esta incluye los siguientes aspectos:

1. Si la RSI (Rutina de Servicio de la Interrupción) necesita un cambio de contexto, el retardo será mayor.

2. La cantidad de tiempo necesaria para realizar la RSI, esto depende en gran parte de la plataforma hardware.

3. El efecto de anidamiento de interrupciones. Si una RSI puede ser interrumpida por la llegada de otra interrupción, entonces el servicio se retrasará.

El determinismo junto a la reactividad conforman el tiempo de respuesta a eventos externos.

El control del usuario es generalmente mucho mayor en un sistema operativo de tiempo real que en SOs ordinarios. En los SOs de tiempo real es necesario permitirle al usuario un control de grano fino sobre la prioridad de las tareas. De esta forma, el usuario debe ser capaz de distinguir entre tareas duras y suaves y de especificar prioridades relativas dentro de cada clase. Un SO de tiempo real puede también permitirle al usuario especificar características como el uso de paginación en los procesos, qué procesos deben residir siempre en memoria principal, qué algoritmo de transferencia a disco deben utilizarse, qué derechos tienen los procesos de las varias bandas de prioridad…

La fiabilidad es normalmente mucho más importante para los sistemas de tiempo real que para los que no lo son.

Un SO de tiempo real debe estar preparado para responder a varios modos de fallo. La operación de fallo suave es una característica que se refiere a la habilidad del sistema de fallar de tal manera que se preserve tanta capacidad y datos como sea posible.

Un aspecto importante de la operación de fallo suave se conoce como estabilidad. Un sistema de

tiempo real es estable si, en los casos en los que sea imposible cumplir los plazos de todas las tareas, este cumplirá los plazos de sus tareas más críticas, aunque no se satisfagan los plazos de algunas tareas.

Para cumplir estos requisitos, los SOs de tiempo real incluyen de forma representativa las siguientes características:

1. Cambio de proceso o hilo rápido.
2. Pequeño tamaño (funcionalidades mínimas).
3. Capacidad para responder rápidamente a interrupciones externas.
4. Multitarea con herramientas para la comunicación entre procesos como semáforos, señales y eventos.
5. Utilización de ficheros secuenciales especiales que pueden acumular datos a alta velocidad.
6. Planificación expulsiva basada en prioridades.
7. Minimización de los intervalos durante los cuales se deshabilitan las interrupciones.
8. Primitivas para retardar tareas durante una cantidad dada de tiempo y para parar/retomar tareas.
9. Alarmas y temporizaciones especiales.

El corazón de un sistema de tiempo real es el planificador de tareas a corto plazo. En el diseño de tal planificador, la equidad y la minimización del tiempo medio de respuesta no son lo más importante. Lo que es importante es que todas las tareas de tiempo real duro se completen (o comiencen) en su plazo y que tantas tareas de tiempo real suave como sea posible también se completen (o comiencen) en su plazo. La mayoría de los sistemas operativos de tiempo real contemporáneos son incapaces de tratar directamente con plazos límite. En cambio, se diseñan para ser tan sensibles como sea posible a las tareas de tiempo real de manera que, cuando se aproxime un plazo de tiempo, la tarea pueda ser planificada rápidamente.

En un planificador expulsivo (apropiativo) que utiliza una simple planificación de turno circular, una tarea de tiempo real sería añadida a la cola de listos para esperar su próxima rodaja de tiempo. En este caso, el tiempo de planificación será generalmente inaceptable para aplicaciones de tiempo real. De modo alternativo, con un planificador no expulsivo (no apropiativo), puede utilizarse un mecanismo de planificación por prioridad, dándole a las tareas de tiempo real mayor prioridad. En este caso, una tarea de tiempo real que esté lista será planificada tan pronto el proceso actual se bloquee o concluya. Esto podría dar lugar a un retardo de varios segundos si una tarea lenta de baja prioridad estuviera ejecutando en un momento crítico. Una vez más, este enfoque no es aceptable. Un enfoque más prometedor es combinar prioridades con interrupciones basadas en el reloj. Los puntos de expulsión ocurrirán a intervalos regulares. Cuando suceda un punto de expulsión, la tarea actualmente en ejecución será expulsada si hay esperando una tarea de mayor prioridad. Esto podría incluir la expulsión de tareas que son parte del núcleo del sistema operativo. Este retardo puede ser del orden de varios milisegundos. Mientras este último enfoque puede ser adecuado para algunas aplicaciones de tiempo real, no será suficiente para aplicaciones más exigentes. En aquellos casos, la opción que se ha tomado se denomina a veces como expulsión inmediata. En este caso, el sistema operativo responde a una interrupción casi inmediatamente, a menos que el sistema esté en una sección bloqueada de código crítico. Los retardos de planificación para tareas de tiempo real pueden reducirse a 100 ms o menos.

Sistemas Operativos Distribuidos y de Red: Stalling pp. 596-599

Página 596

Junto con la creciente disponibilidad de computadores personales asequibles y potentes, ha habido

una tendencia hacia el proceso de datos distribuido (DDP), en el que los procesadores, los datos y el procesamiento de datos pueden estar diseminados por toda la organización. Un sistema DDP implica dividir las funciones de computación y organizar de forma distribuida las bases de datos, el control de dispositivos y el control de interacciones.

En muchas organizaciones, existe una gran dependencia de los computadores personales y su asociación con los servidores. Se necesita una conexión entre los computadores personales y entre los computadores personales y los servidores.

Se han explorado diversas soluciones distribuidas:

- **Arquitectura de comunicaciones:** Es un software que da soporte a un grupo de computadores en red. Proporciona soporte para aplicaciones distribuidas, tales como correo electrónico, transferencia de ficheros y acceso a terminales remotos. Sin embargo, los computadores siguen siendo entidades independientes para los usuarios y para las aplicaciones, que se deben comunicar entre sí por expreso deseo. Cada computador tiene su propio sistema operativo y es posible tener una mezcla de computadores y sistemas operativos, siempre y cuando todas las máquinas soporten la misma arquitectura de comunicaciones. La arquitectura de comunicaciones más ampliamente utilizada es el conjunto de protocolos TCP/IP, que se examina en este capítulo.

- **Sistema operativo de red:** En esta configuración hay una red de máquinas, normalmente estaciones de trabajo de un solo usuario, y una o más máquinas servidoras. Éstas proporcionan servicios de red o aplicaciones, tales como almacenamiento de ficheros y gestión de impresión. Cada computador tiene su propio sistema operativo. El sistema operativo de red es un añadido al sistema operativo local, que permite a las máquinas interactuar con los servidores. El usuario conoce la existencia de múltiples computadores y debe trabajar con ellos de forma explícita. Normalmente se utiliza una arquitectura de comunicaciones común para dar soporte a estas aplicaciones de red.

- **Sistema operativo distribuido.** Es un sistema operativo común compartido por una red de computadores. A los usuarios les parece un sistema operativo normal centralizado, pero les proporciona acceso transparente a los recursos de diversas máquinas. Un sistema operativo distribuido puede depender de una arquitectura de comunicaciones para las funciones básicas de comunicación, pero normalmente se incorporan un conjunto de funciones de comunicación más sencillas para proporcionar mayor eficiencia.

La tecnología de la arquitectura de comunicaciones está bien desarrollada y es soportada por todos los vendedores. Los sistemas operativos de red son un fenómeno más reciente, pero existen algunos productos comerciales. La parte central de investigación y desarrollo de sistemas distribuidos se centra en el área de sistemas operativos distribuidos.

Cuando un computador, terminal y/u otro dispositivo de procesamiento de datos intercambia datos, el procedimiento involucrado puede ser muy complejo.

Hace falta que el emisor active la comunicación con el otro extremo, que se verifique si el receptor está recibido para recibir datos y que se compruebe si el formato de los datos es compatible con el receptor, en caso contrario se deberán traducir.

Para todo esto se emplean los protocolos, estos se usan para comunicar entidades de diferentes sistemas y tienen los siguientes elementos principales:

- Sintaxis. Incluye cosas tales como formato de datos y niveles de señales
- Semántica. Incluye información de control para realizar coordinación y gestión de errores.
- Temporización. Incluye ajuste de velocidades y secuenciamiento.

De esta forma, se puede definir una arquitectura de protocolos como un conjunto de protocolos que se emplean para asegurar la comunicación entre 2 terminales. Unos de ellos se encargarán de mantener la integridad de los datos, otros de que asegurar que estos puedan ser transmitidos…

Sistemas Operativos Multiprocesador (o Paralelos): Stalling pp. 172-175

Página 172

ARQUITECTURA SMP

Los dos enfoques más populares para proporcionar paralelismo a través de la réplica de procesadores son el multiprocesamiento simétrico (SMP) y los clusters. Aquí veremos el SMP.

La arquitectura SMP, según la taxonomía de procesamiento paralelos introducida por Flynn, encaja la categoría de MIMD (Multiple Instrucion Multiple Data).

Con la organización MIMD, los procesadores son de propósito general, porque deben ser capaces de procesar todas las instrucciones necesarias para realizar las transformaciones de datos apropiadas. MIMD se puede subdividir por la forma en que se comunican los procesadores. Si cada procesador tiene memoria dedicada, cada elemento de proceso es en sí un computador. La comunicación entre computadores se puede realizar a través de rutas prefijadas o bien a través de redes. Este sistema es conocido como cluster o multicomputador. En cambio, si los procesadores comparten memoria común, cada procesador accede a los programas y los datos almacenados en la memoria compartida, y los procesadores se comunican entre sí a través de dicha memoria. Este sistema se conoce como multiprocesador de memoria compartida.

Estos, a su vez, se pueden dividir en maestro/esclavo y simétricos. Los maestro/esclavo son el típico que el maestro es un procesador que ejecuta el núcleo del SO, mientras que el resto de procesadores solo ejecutan programas de usuario y algunas utilidades del SO. El maestro es responsable de la planificación de hilos (en resumen, esto funciona como una arquitectura cliente/servidor). La resolución de conflictos se simplifica puesto que un procesador tiene el control de toda la memoria y recursos de E/S. Las desventajas son que si el maestro se cae, falla el sistema entero y que puede crearse un cuello de botella en el maestro.

En un multiprocesador simétrico el núcleo puede ejecutarse en cualquier procesador, y normalmente cada procesador realiza su propia función de planificación del conjunto disponible de procesos e hilos. El núcleo puede construirse como múltiples procesos o múltiples hilos permitiéndose la ejecución de partes del núcleo en paralelo. El enfoque SMP complica al sistema operativo, ya que debe asegurar que dos procesadores no seleccionan un mismo proceso y que no se pierde ningún proceso de la cola. Se deben emplear técnicas para resolver y sincronizar el uso de los recursos.

Página 174

ORGANIZACIÓN SMP

Existen múltiples procesadores, cada uno de los cuales contiene su propia unidad de control, unidad aritmético-lógica y registros. Cada procesador tiene acceso a una memoria principal compartida y dispositivos de E/S a través de algún mecanismo de interconexión; el bus compartido es común a todos los procesadores. Los procesadores se pueden comunicar entre sí a través de la memoria (mensajes e información de estado dejados en espacios de memoria compartidos). Los procesadores han de poder intercambiarse señales directamente. A menudo la memoria está organizada de tal manera que se pueden realizar múltiples accesos simultáneos a bloques separados.

En máquinas modernas, los procesadores suelen tener al menos un nivel de memoria cache, que es privada para el procesador. El uso de esta cache introduce nuevas consideraciones de diseño. Debido a que la cache local contiene la imagen de una porción de memoria principal, si se altera una palabra en una cache, se podría invalidar una palabra en el resto de las caches. Para prevenir esto, el resto de los procesadores deben ser alertados de que se ha llevado a cabo una actualización. Este problema se conoce como el problema de coherencia de caches y se suele solucionar con técnicas hardware más que con el sistema operativo.

CONSIDERACIONES DE DISEÑO DE SOs MULTIPROCESADOR

Un sistema operativo SMP gestiona los procesadores y otros recursos del computador, de manera que el usuario puede ver al sistema de la misma forma que si fuera un sistema uniprocesador multiprogramado.Un usuario puede desarrollar aplicaciones que utilicen múltiples procesos o múltiples hilos dentro de procesos sin preocuparse de si estará disponible un único procesador o múltiples procesadores. De esta forma, un sistema operativo multiprocesador debe proporcionar toda la funcionalidad de un sistema multiprogramado, además de características adicionales para adecuarse a múltiples procesadores. Las principales claves de diseño incluyen las siguientes características:

- **Procesos o hilos simultáneos concurrentes.** Las rutinas del núcleo necesitan ser reentrantes para permitir que varios procesadores ejecuten el mismo código del núcleo simultáneamente. Debido a que múltiples procesadores pueden ejecutar la misma o diferentes partes del código del núcleo, las tablas y la gestión de las estructuras del núcleo deben ser gestionas apropiadamente para impedir interbloqueos u operaciones inválidas.

- **Planificación.** La planificación se puede realizar por cualquier procesador, por lo que se deben evitar los conflictos. Si se utiliza multihilo a nivel de núcleo, existe la posibilidad de planificar múltiples hilos del mismo proceso simultáneamente en múltiples procesadores.

- **Sincronización.** Con múltiples procesos activos, que pueden acceder a espacios de direcciones compartidas o recursos compartidos de E/S, se debe tener cuidado en proporcionar una sincronización eficaz. La sincronización es un servicio que fuerza la exclusión mutua y el orden de los eventos. Un mecanismo común de sincronización que se utiliza en los sistemas operativos multiprocesador son los cerrojos.

- **Gestión de memoria.** La gestión de memoria en un multiprocesador debe tratar con todos los aspectos encontrados en las máquinas uniprocesador. Además, el sistema operativo necesita explotar el paralelismo hardware existente, como las memorias multipuerto, para lograr el mejor rendimiento.

- **Fiabilidad y tolerancia a fallos.** El sistema operativo no se debe degradar en caso de fallo de un procesador. El planificador y otras partes del sistema operativo deben darse cuenta de la pérdida de un procesador y reestructurar las tablas de gestión apropiadamente.