Sistemas operativos II

Temario:

1. Sistemas operativos en ambientes distribuidos.
2. Comunicación en los S.O. distribuidos.
3. Procesos y procesadores en los S.O.D.
4. Memoria compartida distribuida.
5. Usos y tendencias de los S.O.D.

Examen diagnóstico.

1. ¿Qué es un sistema operativo?

Es aquel que gestiona un conjunto de drivers para los dispositivos y al igual software o programas para la visualización del usuario.

1. ¿Qué es un proceso?

Son un conjunto de instrucciones ordenadas que se ejecutan de manera ordenada o simultanea.

1. ¿Qué es un hilo de ejecución?

Es un proceso ejecutado mientras que otro es ejecutado y es atendido entre lapsos de tiempo para liberación de memoria.

1. ¿Cómo funciona la memoria virtual?

Compaginación y almacenamiento de la información segmentada temporalmente.

1. ¿Qué es una llamada al sistema?

Son aquellas que hacen posible la ejecución de cada proceso y dejan en segundo plano el otro proceso.

1. ¿Qué es exclusión mutua?

Cuando un proceso es primordial, este se a dueña de los recursos y no permite que otro proceso entre.

1. Que significa RPC?

Llamadas de procesos remotos

1. ¿Cuáles son las capas del modelo OSI?

Física, enlace, red, transporte, presentación, capa de red, aplicación

1. Elabora un diagrama de una red cliente/servidor.

Cliente -🡪ethertnet-🡪servidor-🡪BD

¿Qué es un S.O.D?

Es una colección de computadoras independientes que aparecen ante los usuarios como una única computadora.

Una de las principales razones para construir S.O es que ciertas aplicaciones son distribuidas de forma inherente, por ejemplo los servidores DNS indispensables para el funcionamiento de las direcciones de internet.

Ejemplo

* Banco sucursales en todo el mundo.
* Venta de boletos de avión.
* Etc.

Ventajas respecto a un sistema centralizado.

* Escalabilidad: crecimiento gradual del sistema.
* Flexibilidad: utilización de máquinas de diversas arquitecturas.
* Disponibilidad: mediante replicación de recursos ofrecen posibilidades al 99.99%
* Permiten acceder a recursos remotos.

Ventajas respecto a un sistema de red.

* Uso más eficiente de los recursos de red.
* Acceso transparente a los recursos.

Desventajas respecto a un sistema.

* Si la distribución de recursos es inadecuada algunos recursos pueden estar desbordados mientras otros están libres.
* Mantener la consistencia del sistema puede ser muy costoso.
* Los algoritmos de control son más complejos y difíciles de implementar.

Tendencias.

* Informática móvil.
* Nuevos dispositivos: PDAS, teléfonos móviles.
* Redes inalámbricas.
* Sistemas ubicuos (pervasive systems).
* Computación ubicua: hogar.
* La gestión de la seguridad es más compleja.

Transparencia.

* Identificación.
* Ubicación.
* Replicación.
* Compartición.

Estabilidad:

Capacidad de crecer sin disminuir su rendimiento.  
Basada en la modularidad y en los espacios de nombres que por lo general son jerárquicos y escalables.

Fiabilidad:

Capacidad para realizar correctamente y en todo momento las funciones para las que se ha diseñado.

Disponibilidad:

Fracción de tiempo que él está.

Tolerancia a fallos:

Capacidad para seguir operando correctamente ante el fallo de alguno de sus componentes.

Consistencia.

* Problemas relacionados con la replicación.
* La red de interconexiones es una nueva fuente de fallos.
* La seguridad del sistema es más vulnerable.
* La gestión del estado global es más compleja y costosa.

Problemas para mantener la consistencia.

* Distribución física: existen varias copias, cada una con diversas modulaciones.
* Errores y/o retardos en las comunicaciones
* Ausencia de reloj global ¿Cómo ordenar eventos?

Categorías o taxonomía de flym para los sistemas distribución.

SISD - Single Instruction, single Data. – todas las computadoras de 1 procesador

SIND – Singles instruction, multiple data. – mismo cálculo para múltiples datos

MISD – multiple instruction single data – ninguna

MIMD – multiple instruction, multiple data – sistemas distribuidos.

|  |  |  |
| --- | --- | --- |
| Instrucción | 1 | Datos |
| Instrucción | 2 | Datos |
| Instrucción | 3 | Datos |

Ventajas de los sistemas distribuidos con respecto de los centralizados.

1. EL que ciertas aplicaciones sean distribuidas de forma inherente.

Es amplio para que denote cualquier sistema en el que varios CPU conectados entre si trabajen de manera conjunta en ciertas puntos dentro de una computadora local.

1. Al trabajo cooperativo apoyado por computadora.

Es el conjunto de usuarios que están a ciertas distancias lejanas y se unen para trabajar juntos y producir un informe en conjunto.

1. Juegos cooperativos apoyados por computadora.

Es cuando los jugadores, se unen a un host y pueden jugar un mismo juego en tiempo real y estadísticas que se actualizan en tiempo real. Incluso las gráficas se actualizan en tiempo real de como los jugadores están ubicados en ciertos lugares y otros en otros.

1. Una mayor confiabilidad que un sistema centralizado.

Es confiable ya que el trabajo es distribuido a muchas maquinas lo cual si una se llegara a suspender o caer, hay un backup de la información y este puede seguir continuando con los demás usuarios sin afectar el trabajo o la información.

1. El crecimiento por incrementos.

Tiene una gran ventaja ya que no genera costos a largo plazo, ya que este permite trabajar en pequeñas magnitudes y grandes magnitudes y para mayor eficacia se pueden utilizar más procesadores, sin en cambio en un mainframe no, viene más limitado y es costoso y no es escalable.

1. La flexibilidad potencial frente al hecho de darle.

Es bastante confortable ya que cada usuario tiene su propio ordenador y privacidad que permite que no le interrumpa otro usuario, y este puede trabajar independientemente en equipo desde cualquier punto en un informe en común sin que invadan su privacidad.

Desventajas de los sistemas distribuidos.

La primera desventaja es el diseño del software y diseño, ya que es muy complicado gestionar varios operadores en un sistema operativo distribuido.

Segunda desventaja es a las redes de comunicación, cuando se sobre carga el sistema se suele perder los mensajes y dejar a los usuarios incomunicados por saturación de red.

Conceptos de HARDWARE.

SISD (SINGLE INSTRUCTION SINGLE DATA) computadoras tradicionales de un procesador como computadoras personales hasta las grandes mainframes.

SIMD (SINGLE INSTRUCTION MULTIPLE DATA) Flujo de instrucciones y varios flujos de datos. Procesadores con una unidad de instrucciones que busca una instrucción para que la lleve a cabo en forma paralela cada uno de sus propios datos.

MISD (MULTIPLE INSTRUCTION SINGLE DATA) Flujo de varias instrucciones y un flujo de datos.

MIMD contiene memoria compartida por lo general se llama multiprocesadores o multi-computadoras casi no es muy útil para nuestros fines.

30/01/15

Como se ha visto en la taxonomía del flynn todos los sistemas distribuidos corresponden a la categoría MIMD.

La categoría MIMD puede dividirse a su vez en dos grupos:

Aquellos sistemas que comparten memoria entre sí (multiprocesadores).

Aquellos sistemas en que cada CPU tiene su propia memoria (multicomputadoras).

Cada una de estas subcategorías se pueden a su vez dividir en base en la arquitectura de interconexión:

Basadas en bus: los dispositivos están conectados por un bus (cableado de “columna vertebral”).

Basadas en conmutador: los dispositivos utilizan cableados individuales hacia un conmutador (switch) que dirige los mensajes hacia el destino por la salida que corresponden respectivamente.

Otra clasificación consiste en que ciertos sistemas son “fuertemente acoplados” y otros están “débilmente acoplados”.

En un sistema fuertemente acoplado el retraso al enviar un mensaje entre dispositivos es corto, la tasa de trasmisión de dato (número de bits por segundo) es alta. En un sistema débilmente acoplado ocurre lo contrario.

Dos cpu conectados en la misma “tarjeta madre” o placa base es un sistema fuertemente acoplados.

Dos computadoras conectadas por medio de un modem de 2400 bits por segundo están débilmente acopladas.

Los multiprocesadores tienden a estar más fuertemente acoplados que las multicomputadoras puesto que pueden intercambiar datos a mayor velocidad.

Sin embargo algunas multicomputadoras conectadas con fibra óptica pueden funcionar también a esa velocidad.

04/02/15

Si en un sistema distribuido el hardware es importante, el software lo es aún más.

De manera análoga que en el hardware, el software también se clasifica en débil o débil o fuertemente acoplado.

Software débilmente acoplado: permite que las máquinas y los usuarios sean independientes entre si pero que interactúen en cierto grado cuando es necesario.

Por ejemplo un grupo de computadoras personales conectadas es una red local, cada una cuenta con su propio procesador, memoria, disco duro pero comparten algunos dispositivos como impresoras.

Esta débilmente acoplado puesto que cada computadora tiene sus propias tareas por realizar, si la red falla cada una continúa operando.

En el otro extremo considere el caso de tener un multiprocesador (varios procesadores en la misma tarjeta base) dedicados a la ejecución de un programa de ajedrez en paralelo.

A cada procesador se le asigna una porción del cálculo de las posibilidades y al terminar informa del resultado y se le asigna un nuevo cálculo que realizar.

El software de este sistema, es decir, el programa del juego y el sistema operativo están fuertemente acoplados.

Sistemas operativos de redes:

En este modelo cada usuario tiene una computadora para su uso exclusivo. Tiene su propio sistema operativo. Las tareas se ejecutan de forma local.

Es posible que el usuario se conecte con otra computadora de manera remota (escritorio remoto), lo que convierte la propia estación de trabajo en un terminal.

Las entradas del teclado y los movimientos del mouse se envían a la maquina remota y se exhiben en aquel monitor.

También se cuenta con comandos y protocolos para copiado remoto (FTP por ejemplo) pero esto funciona con la intervención explicita del usuario que además se requiere conocer la posición de los archivos.

Servidores de archivos:

Proporcionan un sistema de archivos global, compartido, accesible desde todas las terminales de trabajo.

Aceptan solicitudes para la lectura y escritura de archivos de parte de los programas de usuario que se ejecutan en otras máquinas (llamadas clientes).

Cada una de las solicitudes que llega se examina, se ejecuta y la respuesta se envía de regreso.

Cuentan con un sistema jerárquico de archivos, con un directorio raíz, subdirectorios y archivos.

El sistema operativo tiene que controlar las peticiones de las estaciones de trabajo, los directorios de archivos y de la comunicación.

Sistemas realmente distribuidos:

El objetivo de un sistema distribuido es crear la ilusión que toda la red de computadoras es un sistema compartido en vez de una colección de máquinas diversas.

Un sistema distribuido es aquel que se ejecuta en una colección de máquinas entrelazadas en una red pero que actúan como un “uniprocesador” virtual.

¿Cuáles son las características de un sistema distribuido?

Debe de existir un mecanismo de comunicación global entre procesos.

Tiene que haber un mismo mecanismo de comunicación local y remota en cada máquina.

Debe de existir un esquema global de protección.

La administración de procesos debe ser la misma en todas las máquinas.

Ningún sistema en la actualidad cumple con ese requisito.

Actividad.

Transparencia de la localización.

De que un sistema distribuido, los usuarios no puede indicar la localización de los recursos de hardware y software, como los CPU, impresoras, archivos y bases de datos.

Transparencia de migración

Los recursos de deben moverse de una posición a otra sin tener que cambiar sus nombres. La forma de los directorios servidores se podían montar en lugares arbitrarios dentro de la jerarquía de directorios de los clientes.

Transparencia replica.

Entonces el sistema operativo es libre de fabricar por su cuenta copias adicionales de los archivos y otros recursos sin que lo noten los usuarios. Consideremos de n servidores, conectados de manera lógica para formar un anillo.

Transparencia con respecto a la concurrencia

Los usuarios no notaran la existencia de otros usuarios. Un mecanismo para lograr esta forma de transparencia seria que el usuario cerrara en forma automática un recurso, una vez que alguien haya comenzado a utilizarlo, elimina el bloqueo solo hasta que termine el acceso.

Transparencia con respecto al paralelismo

En principio, se supone que un sistema distribuido debe aparece ante los usuarios como un sistema tradicional de tiempo compartido con un procesador. Es que el compilador, el sistema de tiempo de ejecución y el sistema operativo deben en forma conjunta aprovechar este paralelismo potencial sin que el programador se dé cuenta de ello.

Flexibilidad.

Es importante que sea flexible, es probable que este proceso tenga muchas salidas falsas y un considerable retroalimentación. Las decisiones de diseño que ahora parezcan razonables podrían demostrar ser incorrectas posteriormente. La mejor forma de evitar los problemas es mantener abiertas las opciones.

Confiabilidad.

Uno de los objetivos originales de la construcción de sistemas distribuidos fue el hacerlos más confiables que los sistemas de procesador. La idea es que si una maquina falla, alguna otra máquina se encargue del trabajo. En otras palabras es teoría, la confiabilidad global del sistema podría ser el OR booleano de la confiabilidad de los componentes.

Disponibilidad como acabamos de ver, se refiere a la fracción del tiempo en que se puede utilizar el sistema. El sistema de lamport no tiene buena calificación a este respecto. La disponibilidad, puede mejorar mediante un diseño que no exija el funcionamiento simultáneo de un número sustancial de componentes críticos.

Desempeño

La construcción de un sistema distribuido transparente flexible y confiable no hará que usted gane premios si es tan lento como la miel. Particularmente cuando se ejecuta una aplicación en un sistema distribuido, no debe parecer peor que su ejecución en un procesador. Por desgracia, esto es más difícil de lograr.

El problema de desempeño se complica por el hecho de la comunicación, factor esencial en un sistema distribuido.

Escalabilidad.

La mayor parte de los sistemas distribuidos están diseñados para trabajar con unos cuantos cientos de cpu. Es posible que los sistemas futuros tengan mayores órdenes de magnitud y las soluciones que funcionen bien para 200 máquinas fallen en manera total.

Los sistemas deben contar con escalabilidad para que puedan a futuros funcionar con futuros hardware y poder funcionar a futuras generaciones grandes magnitudes.

Sistemas operativos distribuidos.

Amoeba es un sistema distribuido.

Este permite que una colección de CPU y equipo en ent/sal se comporten como una única computadora.

También proporciona elementos para la programación paralela.

Amoeba se originó en una universidad de Ámsterdam como un proyecto de investigación en cómputo distribuido y paralelo.

Muchos de los proyectos de investigación en los sistemas operativos distribuidos han partidos de un sistema existen (por ejemplo UNIX) al que le han agregado nuevas características de uso de redes y sistemas compartidos de archivos.

El proyecto Amoeba siguió un método diferente y desarrollo un nuevo sistema a partir de cero.

La idea fue tener un inicio en limpio para no tener que preocuparse por la compatibilidad retroactiva con los componentes del sistema base.

Para no tener que escribir todo el software de aplicación se añadió un paquete de emulación de UNIX.

El objetivo principal fue construir un sistema operativo distribuido transparente que para que los usuarios lo usen como un sistema de tiempo compartido en el que se pueden entre otras cosas ejecutar programas y almacenar archivos.

La diferencia estaba en que estas acciones en amoeba se realizan en varis computadoras dispersas en la red. Entre esas computadoras están los servidores de procesos y servidores de archivos, pero el usuario no es consciente de ello.

En amoeba no se tiene el concepto de computadora de origen. Cuando un usuario entra al sistema, entra a este como un todo y no a una computadora específica, ya que estas no tienen propietarios.

El intérprete de comandos que se ejecuta al entrar el usuario se ejecutan en una computadora arbitraria y los procesos o tareas requeridos por el usuario no se ejecutan precisamente sobre esa misma computadora del intérprete de comandos.

Amoeba busca de manera automática la computadora con la menor carga para ejecutar de nuevo cada comando.

Con el transcurrir de la sesión del usuario sus procesos estarán esparcidos por casi todas las computadoras del sistema.

En otras palabras, todos los recursos pertenecen al sistema como un todo y son controlados por el mismo.

Amoeba es transparente con respecto con la ubicación. Ya que es el sistema y no el usuario quien determina en que computadora debe ejecutarse los procesos sin que esto sea visible para el usuario.

El segundo objetivo de amoeba es proporcionar una estructura de prueba para la realización de experimentación con algoritmos, lenguajes y aplicaciones distribuidas paralelas.

Amoeba ofrece soporte a investigaciones por medio de un lenguaje de nombre orca que fue diseñado e implementado en el sistema.

09/02/2015

MACH

Historia y objetivos.

Las raíces de mach van hasta un sistema llamado TIG (Rochester intelligent Gateway) que se inició en la universidad de Rochester en 1975.

Fue escrito para una computadora en 16 bits llamada eclipse.

Su principal objetivo de investigación era demostrar que podían estructuras los sistemas operativos de manera modular, como una colección de procesos que se comuniquen entre si mediante la transferencia de mensajes, incluso a través de la red.

Uno de los colaboradores de este proyecto se trasladó a otra universidad (1979) con el deseo de continuar desarrollando sistemas operativos con transferencia de mensajes pero en un hardware más potente.

Este nuevo sistema operativo fue perdiendo preferencia ante la proliferación de UNIX. Esto propicio que se iniciara el proyecto de un tercer sistema operativo que se llamaría mach.

Se esperaba, que al hacerlo compatible con unix podría utilizarse todo el software que para él se venía desarrollando.

Mach tenía mejorar con respecto a Accent en los hilos de ejecución, mejores mecanismos de comunicación entre procesos, soporte multiprocesador y un sistema de memoria virtual.

Mach continúo evolucionando sobre todo con el patrocinio del departamento de defensa de EU que requería de un sistema operativo que soportara multiprocesadores como parte de un programa de “computo estratégico”.

Se desarrollaron versiones para computadoras de diferentes arquitecturas tales como la vax 11/748 o la IBM PC/RT y la SUN 3

Mach continuo desarrollándose, ha remplazado su núcleo original (que era grande y monolítico), los objetivos originales del proyecto también han ido cambiando.

Los actuales objetivos de mach se pueden resumir de la siguiente manera:

Proporcionar una base para construir otros sistemas operativos.

Permitir el acceso transparente a los recursos de la red.

Explotar el paralelismo tanto en el sistema operativo como por las aplicaciones.

Lograr que mach se pueda transportar a un sistema de mayor número de máquinas.

Explorar los multiprocesadores y los sistemas distribuidos a la vez que se puedan emular los sistemas existentes (UNIX por ejemplo)

Cuando mach se diseñó pocos sistemas operativos tenían soporte para multiprocesadores.

Aun ahora existen pocos sistemas para multiprocesadores que además sean independientes de la arquitectura de una computadora específica.

CHORUS

Ha evolucionado a lo largo de los años, analizaremos sus objetivos de forma breve y después daremos una introducción técnica a su micro núcleo y sub-sistemas. En las secciones posteriores describiremos una terminología poco estándar.

Surgió en el instituto francés de investigación INRIA en 1980, como proyecto de investigación en sistemas distribuidos.

La versión 1 se utilizó 82 a 84 se centró en la investigación del multi-procesador. Un CPU ejecutaba UNIX; las otras siete ejecutaban chorus y utilizan el CPU de unix para los servicios del sistema y e/s.

La versión 2 84/86 fue una reescritura fundamental del sistema en C. Se diseñó de modo que las llamadas al sistema fuesen compatibles con UNIX en el nivel del código fuente lo que significa que pueden recopilar los programas existentes en UNIX en chorus y ejecutarlos en el.

La versión 3 inicio en 1987 marco la transición de un sistema de investigación a un producto comercial, ya que los diseñadores de chorus salieron de INRIA y formaron una compañía.

Objetivos.

En un principio, se trataba de una investigación puramente académica, diseñada para explorar nueva ideas en el computo distribuido con base en el modelo actor.

1. Emulación de UNIX de alto rendimiento.
2. Uso en sistemas distribuidos.
3. Aplicaciones de tiempo real.
4. Integración de la programación orientada a objetos en chorus.

Un segundo tema fundamental es la necesidad de la distribución. Otras como la introducción de hils, requieren repensar las características existentes, como el manejo de las señales de unix.

Una tercera dirección es la introducción de un soporte para las aplicaciones de tiempo real. El enfoque en este caso permite que los programas de tiempo real se ejecuten, en modo núcleo y que tengan acceso directo al micronucleo, sin software de por medio.

RMI (Java **Remote Method Invocation**) es un mecanismo ofrecido por Java para invocar un método de manera remota.

SOCKET Los **sockets** son básicamente formas en las que podemos interconectar 2 (o mas) programas mediante el uso de la internet.

CORBA **Common Object Request Broker Architecture** es un estándar definido por Object Management Group (OMG) que permite que diversos componentes de software escritos en múltiples lenguajes de programación y que corren en diferentes computadoras, puedan trabajar juntos; es decir, facilita el desarrollo de aplicaciones distribuidas en entornos heterogéneos.

DCOM puede referirse a: Distributed Component Object Model, un sistema de Microsoft.