Algoritmos de planificación del procesador en entornos distribuidos

Sistemas de Operación II

Prof. Rodolfo Campos
rcampos@ucab.edu.ve

Universidad Católica Andrés Bello
Caracas, 2006

Modelos de Sistemas

Modelo de estaciones de trabajo

Si la estación tiene sus propios discos estos se pueden utilizar para:

- Paginación y archivos temporales.
- Paginación, archivos temporales y binarios del sistema.
- Paginación, archivos temporales, binarios del sistema y sistema de ocultamiento de archivos.
- Sistema local de archivos.

Modelos de Sistemas

Uso de estaciones de trabajo inactivas

Problemas clave:

- ¿Cómo encontrar una estación de trabajo inactiva?. Luego, ¿Qué es una estación de trabajo inactiva? – Algoritmos controlados por el cliente y el servidor
- ¿Cómo lograr que un proceso remoto se ejecute en forma transparente?
- ¿Qué ocurre si regresa el dueño de la máquina?

¿Qué tal una pila de procesadores?
¿Y un híbrido?

Asignación de procesadores

Estrategias de asignación de procesadores

- Asignación migratoria: un proceso se puede trasladar (código, datos del usuario y estructuras del núcleo) así haya iniciado su ejecución en otro procesador.
- Asignación no migratoria: al crearse un proceso se toma una decisión acerca de donde colocarlo, una vez colocado en una máquina el proceso permanece allí hasta que termina.

Asignación de procesadores

Acerca del diseño

Algoritmos deterministas vs. heurísticos.

Los algoritmos deterministas son adecuados cuando se conoce todo acerca del comportamiento de los procesos. En teoría, se podrían intentar todas las posibles asignaciones y tomar la mejor.

En la práctica son muy pocos los escenarios en los que se conocen todos los datos por lo que se requiere la implementación de algoritmos heurísticos (*ad hoc*).

Asignación de procesadores

Acerca del diseño

Algoritmos centralizados vs. distribuidos.

"La recolección de toda la información en un lugar permite tomar una mejor decisión, pero menos robusta y coloca una carga pesada en la máquina central" – Tanenbaum, 1996

Asignación de procesadores

Acerca del diseño

Algoritmos óptimos vs. subóptimos

Los algoritmos óptimos suelen ser muy costosos computacionalmente y a veces sólo se requiere una buena solución y no la mejor.

Asignación de procesadores

Acerca del diseño

Algoritmos locales vs. globales

Los algoritmos locales toman decisiones basándose en información local, mientras que los globales llevan el estado de todo el sistema.

"Los algoritmos locales son sencillos, pero están muy lejos de ser óptimos, mientras que los globales sólo dan resultados un poco mejores a costos mayores" – Tanenbaum, 1996

Asignación de procesadores

Acerca del diseño

Algoritmos iniciados por el emisor vs. iniciados por el receptor.

El emisor puede decidir que tiene demasiado trabajo y decide enviarlo a un receptor (iniciado por el emisor) o el receptor al verse ocioso decide comunicarle a sus compañeros que está disponible (iniciado por el receptor).

Asignación de procesadores

Acerca de la implantación

Las siguientes son consideraciones importantes:

- Casi todos los algoritmos suponen que las máquinas conocen su carga de trabajo; si están subcargados o sobrecargados.
- Costos de asignación de procesos (recolección de medidas y desplazamiento de procesos).
- Complejidad de los algoritmos utilizados.
- Estabilidad entre máquinas del sistema.

Algoritmos de asignación

Un algoritmo determinista según la teoría de gráficas

- El sistema se puede representar como una gráfica con pesos, donde cada nodo es un procesos y cada arco representa el flujo de mensajes entre dos procesos.
- El problema se reduce a encontrar una forma de partir la gráfica en subgráficas ajenas, sujetas a ciertas restricciones.
- Los arcos contenidos dentro de una subgráfica representan la comunicación entre procesos (descartable) y los arcos que van de una subgráfica a otra representan el tráfico en la red.

Algoritmos de asignación

Algoritmo arriba-abajo (Mutka y Livny, 1987)

- Un coordinador mantiene una **tabla de uso** con una entrada (incializada en 0) por cada estación de trabajo personal.
- En lugar de intentar maximizarse el uso de cada CPU, la preocupación es darle a cada procesador una carga justa de trabajo.
- Cuando se va a crear un proceso y la máquina donde se crea decide que el proceso debe ejecutarse en otra parte, le pide al coordinador que le asigne un procesador. Si no existen procesadores libres, la solicitud se niega por el momento y se toma nota de ella.
- Cuando un procesador ejecuta procesos en otros procesadores, acumula puntos de penalización.

Algoritmos de asignación

Algoritmo arriba-abajo (Mutka y Livny, 1987)

- Las entradas de un procesador en la **tabla de uso** pueden ser positivas, cuando éste es un usuario de los recursos del sistema, cero (neutro) o negativa, cuando éste necesita recursos del sistema.
- Cuando un procesador se libera, gana la solicitud pendiente cuyo poseedor tenga la puntuación más baja en la **tabla de uso**.

Algoritmos de asignación

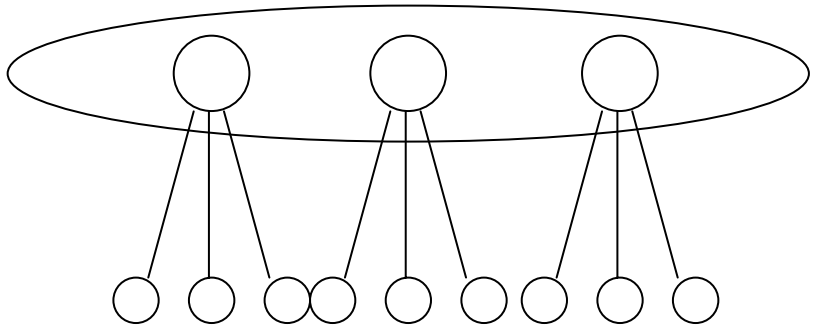
Un algoritmo jerárquico MICROS (Wittie y Van Tilborg, 1980)

- Se organizan las máquinas en jerarquías y algunas de ellas fungen roles de trabajadoras y otras de administradoras.
- A cada grupo de máquinas se les asigna un administrador que mantiene el registro de quienes están ocupadas y quienes no.
- Si un administrador al recibir una solicitud determina que tiene pocos procesadores disponibles, transfiere la solicitud hacia arriba, a su jefe. Si el jefe tampoco la puede manejar, la solicitud se sigue propagando hacia arriba, hasta que alcanza un nivel donde se tiene un número suficiente de trabajadores disponibles.

Algoritmos de asignación

Un algoritmo jerárquico MICROS (Wittie y Van Tilborg, 1980)

- Cuando se localiza un jefe que pueda atender una petición, éste divide la solicitud en partes y las esparce entre los administradores por debajo de él, los cuales a su vez repiten la operación, hasta que la ola de asignación llega al punto inferior, donde se ocupan los procesadores y se propaga hacia arriba del árbol su nuevo estado.



Algoritmos de asignación

Un algoritmo heurístico distribuido iniciado por el emisor

- Al crearse un proceso, la máquina donde se origina envía mensajes de prueba a una máquina elegida al azar, para preguntar si su carga está por debajo de cierto valor. En caso afirmativo, el proceso se envía a ese lugar. Si no se encuentra una máquina adecuada después de N pruebas, el proceso se ejecuta en la máquina de origen.

Hay que observar que bajo condiciones de carga pesada, todas las máquinas enviarán mensajes de prueba a las demás, sobrecargando el sistema.

Algoritmos de asignación

Un algoritmo heurístico distribuido iniciado por el receptor

- El algoritmo del modelo anterior es comenzado por un procesador sobrecargado, en este modelo el algoritmo es iniciado por un procesador subcargado.
- Cuando un procesador se encuentra ocioso elige una máquina al azar y le solicita trabajo. En caso de no encontrar oficio, pregunta N veces a distintas máquinas, si aún no consigue nada, descansa un tiempo y vuelve a comenzar.

Con el algoritmo iniciado por el receptor, cuando el sistema tiene carga pesada, la probabilidad de que una máquina se encuentre ociosa es poca. Por lo que habrán menos mensajes en la red.

Algoritmos de asignación

Un algoritmo de remates (Ferguson et al., 1988)

- Los procesos deben comprar tiempo de CPU para realizar su trabajo y los procesadores venden sus ciclos al mejor postor (Ej. el que ofrezca más).
- Cada procesador anuncia su precio aproximado a través de un archivo que el resto pueda leer. El precio es determinado por su velocidad, tamaño de la memoria, presencia de hardware de punto flotante, entre otros.
- Cuando un procesador desea iniciar un proceso hijo, obtiene el mejor candidato (más barato o más rápido o mejor relación costo/beneficio, etc.) de una lista con todos los procesadores que ofrecen el servicio que requiere, genera una oferta menor o mayor al precio anunciado y la envía.
- Los procesadores reúnen toda las ofertas enviadas a ellos y eligen al mejor postor.

Recuperación y tolerancia a fallas

Sistemas de Operación II

Prof. Rodolfo Campos
rcampos@ucab.edu.ve

Universidad Católica Andrés Bello
Caracas, 2006

Fallas de Componentes

"Una falla es un desperfecto, causado tal vez por un error de diseño, un error de fabricación, un error de programación, un daño físico, el deterioro con el curso del tiempo, condiciones ambientales adversas, entradas inesperadas, un error del operador, roedores comiendo parte del sistema y muchas otras causas" – Tanenbaum, 1996

Fallas de Componentes

Clasificación

- Fallas Transitorias: ocurren una vez y luego desaparecen.
- Fallas Intermitentes: desaparecen y reaparecen.
- Fallas Permanentes: continúan existiendo aún después de reparado el componente con el desperfecto.

Fallas de Sistema

Tipos

- Fallas silentes o de detención: ocurren cuando un procesador se detiene y no responde.
- Fallas bizantinas: ocurren cuando un procesador continúa su ejecución ofreciendo respuestas incorrectas.

Sistemas síncronos vs. asíncronos

Cuando se habla de tolerancia a fallas, un sistema síncrono es aquel en donde por cada mensaje enviado se espera (un tiempo finito) por el acuse de recibo, mientras que en los sistemas asíncronos no se esperan respuestas.

Redundancia

Tipos

- Redundancia de la información: se agregan algunos bits para obtener la información original en caso de pérdida, Ej. Código Hamming, códigos turbo.
- Redundancia del tiempo: se puede realizar una misma operación varias veces en el tiempo en caso de necesitarse.
- Redundancia física: se agregan uno o varios equipos adicionales.

Redundancia física

Técnicas

•Replica activa (maestro-maestro): en esta técnica todas las replicas manejan la misma información. La lectura de solicitudes no altera el estado de los servidores, pero la escritura de solicitudes si lo hace. En este esquema es necesario un tiempo global (problema de transacción atómica).

•Replica primaria (maestro-esclavo): en está técnica un servidor es primario y realiza todo el trabajo, mientras que el otro se mantiene idéntico y en caso de fallas toma el control.

Redundancia física

Replica activa vs. primaria

- La replica primaria es más sencilla que la activa durante la operación normal ya que todos los mensajes van a un servidor y no a un grupo.
- La replica primaria no tiene problemas de ordenamiento de los mensajes ya que todos van a la misma máquina.
- La replica primaria necesita menos máquinas que la activa.
- La replica activa puede trabajar correctamente en presencia de fallas bizantinas, mientras que la primaria no.
- En la replica primaria es difícil conocer cuando el servidor primario ha muerto.

Acuerdos en sistemas defectuosos

Algunos sistemas deben acordar decisiones comunes contando con la posible existencia de errores ocasionados por:

- La entrega de mensajes de manera no confiable (Problema de los dos ejércitos – límite en el envío de ACKs)
- La falla de procesadores, silentes o bizantinas (Problema de los Generales Bizantinos)
- La naturaleza del sistema ¿síncrona o asíncrona?. Fischer et al (1985) demostraron que un sistema distribuido con procesadores asíncronos y retrasos no acotados en la transmisión, no puede llegar a un acuerdo aunque sólo un procesador este fallando – no se pueden distinguir los procesadores lentos de los muertos.

Acuerdos en sistemas defectuosos

Solución de Lamport al problema de los Generales Bizantinos

Suposiciones

A1. Todo mensaje es enviado correctamente.

A2. El receptor de todo mensaje sabe quien lo ha enviado.

A3. La ausencia de un mensaje puede ser determinada.

Además, debe existir una manera de calcular el mensaje que se repite con mayor frecuencia (mayoría) y el número de traidores o entes fallando no debe ser mayor a g/3 + 1; siendo g el número de total de generales.

Acuerdos en sistemas defectuosos

Solución de Lamport al problema de los Generales Bizantinos

Condiciones

Dados:

m, número de generales traidores (entes con falla).

n, número total de generales.

Mayoria, una función que me permite hallar la desición de mayor frecuencia, dado un conjunto de valores n-1.

Acuerdos en sistemas defectuosos

Solución de Lamport al problema de los Generales Bizantinos

El Algoritmo

OM(0), m=0

El Comandante envía su valor calculado a todos los Tenientes.

Cada Teniente utiliza el valor que recibe del Comandante o un valor por defecto en caso de no recibir mensaje.

OM(m), m>0

El Comandante envia su valor a cada Teniente.

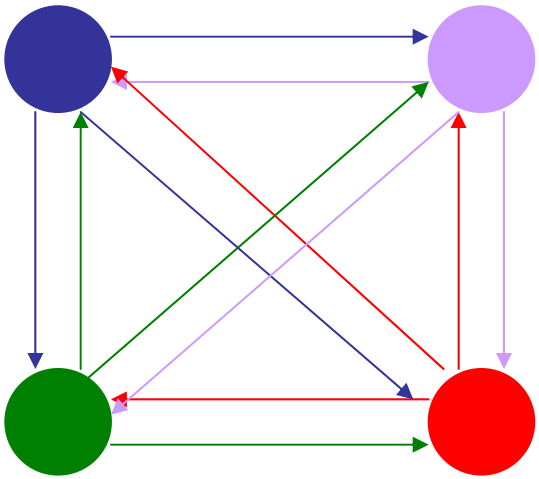
Para cada i, donde Vi es el valor que cada Teniente recibe del Comandante o un valor por defecto en caso de no recibir mensaje. Este Teniente i pasará a actuar como Comandante en el algoritmo OM(m-1), para así enviar el valor Vi a cada uno de los n-2 Tenientes restantes (todos menos él y su Comandante).

Para cada i, y cada j ≠ de i, donde Vj es el valor que el Teniente i recibió del Teniente j en el paso 2 (utilizando el algoritmo OM(m-1)), o un valor por defecto en caso de no recibir mensaje. El Teniente i utilizará el valor Mayoria(V1, ..., Vn-1).

Acuerdos en sistemas defectuosos

Problema de los Generales Bizantinos

Para 4 generales (n) y 1 traidor (m)
El problema tiene solución si existen al menos (2n+1)/3 generales fieles.



Primera iteración: Azul, Morado, Verde y Rojo envían lo que piensan.

Segunda iteración: Azul, Morado, Verde y Rojo envían todas las respuestas que obtuvieron al grupo. Ej. Un vector con lo que dice cada uno del tipo (Azul, Morado, Verde, Rojo)

Tercera iteración: Azul, Morado, Verde y Rojo evalúan los resultados obtenidos, si cualquier valor tiene una mayoría existe una respuesta.

Sistemas distribuidos de tiempo real

Sistemas de Operación II

Prof. Rodolfo Campos
rcampos@ucab.edu.ve

Universidad Católica Andrés Bello
Caracas, 2006

Sistemas de tiempo real

"Los sistemas de tiempo real interactúan con el mundo exterior de una manera que implica al tiempo. Cuando aparece un estímulo, el sistema responde a éste de cierta manera y antes de cierto límite. Si entrega la respuesta correcta, pero después del límite, se considera que el sistema está fallando. **El momento en que se produce la respuesta es tan importante como aquello que la produce**" – Tanenbaum, 1996

Estímulos

- Periódicos. Ocurren de manera regular cada cierto tiempo.
- Aperiódicos. Son recurrentes pero no regulares.
- Esporádicos. Son inesperados.

Sistemas de tiempo real

Clasificación

- Sistemas de tiempo real suave. No existe problema si se rebasa un tiempo límite.
- Sistemas de tiempo real duro. Es inaceptable un tiempo límite no cumplido.

Sistemas de tiempo real

Mitos (Stankovic, 1998)

- Los sistemas de tiempo real tratan de la escritura de controladores de dispositivos en código ensamblador.
- El cómputo de tiempo real es rápido.
- Las computadoras rápidas harán que el sistema de tiempo real sea obsoleto.

Acerca del Diseño

- Sincronización del reloj.
- Sistemas activados por eventos vs. activados por el tiempo.
  - El principal problema con los sistemas activados por eventos es que pueden fallar bajo condiciones de carga pesada. Las lluvias de eventos pueden afectar el sistema.
  - El principal problema con los sistemas activados por el tiempo es que algunos eventos pueden ocurrir entre marcas de reloj. Ahora, operar normalmente en tiempos de crisis aumenta la posibilidad de tratar con éxito la situación.

Acerca del Diseño

- Predictibilidad. Todos deben conocer a priori el orden de las tareas a realizar tras ocurrir un evento.
- Tolerancia a fallas. Se debe enfrentar el número máximo de fallas y la carga al mismo tiempo.
  - La replica activa se utiliza sólo si puede hacerse sin protocolos extensos.
  - La replica primaria no es muy utilizada puesto que los tiempos límite pueden omitirse durante la recuperación.
  - Un método híbrido consiste en seguir al líder, en donde una máquina toma todas las decisiones y las demás realizan su trabajo sin discusión.

Acerca del Diseño

- Soporte del lenguaje.
  - El lenguaje debe diseñarse de modo que se pueda calcular el tiempo máximo de ejecución de cada tarea en tiempo de compilación, por lo que el lenguaje sólo debe soportar ciclos finitos y predefinidos.
  - Los lenguajes de tiempo real necesitan trabajar con el tiempo, por lo que es recomendable disponer de una variable especial reloj (*clock*) de grano proporcional a las capacidades del sistema y el problema.

Comunicación

"Lograr la predictibilidad en un sistema distribuido significa que la comunicación entre los procesadores también debe ser predecible" – Tanenbaum, 1996

Comunicación

- Los protocolos inherentemente estocásticos, como Ethernet, son inaceptables, ya que el azar no es una característica deseada en las soluciones de tiempo real.
- Los protocolos de anillo permiten predecir situaciones a partir de demoras en el tráfico de paquetes en la red, por lo que se adaptan bien a algunas soluciones de tiempo real.

Comunicación

- El protocolo TDMA (*Time division multiple access*) es otra solución que organiza el tráfico en anillos de tamaño fijo de *n* espacios. Cada espacio es asignado a un procesador para enviar y recibir mensajes, evitando así las colisiones, retrasos indefinidos y proveyendo al sistema la capacidad de ofrecer QoS *(Quality of Service).*
- En ocasiones para tratar la pérdida de paquetes, sobre todo en redes WAN, se transmite cada mensaje dos o más veces preferentemente a través de conexiones independientes.

Comunicación

TTP – Time Triggered Protocol (Kopetz y Grunsteidl, 1994)

- Es utilizado en el sistema MARS (Kopetz et al., 1989).
- Un nodo MARS cuenta generalmente con más de un procesador (2 ó 3).
- Los nodos MARS se conectan mediante dos redes de transmisión TDMA confiables e independientes.
- Todos los paquetes se envían por ambas redes en paralelo. La tasa esperada de pérdida de un paquete es de 1 en 30 millones de años.
- El tiempo es crítico y su sincronización es manejada por el mismo protocolo. La precisión puede ser manejada por hardware (dispositivos).

Comunicación

TTP – Time Triggered Protocol (Kopetz y Grunsteidl, 1994)

- Todos los nodos MARS son concientes de los programas que se ejecutan en los demás nodos y pueden detectar los paquetes enviados de un nodos a otro con facilidad.
- Dado que se supone que no se pierden paquetes, la ausencia de un paquete en un momento dado implica la falla de uno de los nodos.

Comunicación

TTP – Time Triggered Protocol (Kopetz y Grunsteidl, 1994)

- Cada nodo MARS mantiene el estado global del sistema dado por:
  - El modo activo. Tiene que ver con la fase en la que se encuentra el sistema. Los modos tienen su propio conjunto de procesos en orden, lista de nodos participantes, asignación de espacios TDMA, nombres y formato de mensajes, y los modos que pueden sucederlo.
  - El tiempo global. Debe ser lo bastante grueso como para que todos los nodos coincidan.
  - Un mapa de bits con la membresía actual del sistema. Registro de los nodos activos e inactivos.

Comunicación

TTP – Time Triggered Protocol (Kopetz y Grunsteidl, 1994)

- Si falta un reconocimiento esperado todos los nodos marcan al emisor como fallido y lo sacan del mapa de bits.
- De forma periódica se transmite un mensaje de iniciación para que aquellos nodos que han quedado fuera puedan unirse como miembros pasivos.
- Cada nodo conoce el tiempo en que inician los marcos TDMA y la posición de su espacio dentro del marco.
- Si un paquete inicia *n* microsegundos antes o después de lo debido, cada uno de los demás nodos puede detectar el problema e intentar corregirlo.

Planificación

Aspectos a tomar en cuenta

- Tiempo real duro vs. tiempo real suave.
- Algoritmos preferentes vs. no preferentes. Los algoritmos preferentes permiten suspender tareas de menor prioridad hasta que no existan más tareas de mayor prioridad.
- Algoritmos dinámicos vs. estáticos. Los algoritmos dinámicos toman decisiones en tiempo de ejecución, mientras que los centralizados no, en éstos cuando ocurre un evento el planificador busca en una tabla predefinida para ver que hacer.
- Algoritmos centralizados vs. descentralizados.

Planificación

Planificación dinámica

Algoritmo monótono de tasa (Liu y Layland, 1973)

- Cada tarea tiene asignada una prioridad igual a su frecuencia de ejecución.
- En tiempo de ejecución el planificador selecciona la tarea de máxima prioridad para su ejecución

Planificación

Planificación dinámica

Algoritmo del primer límite en primer lugar

- Cuando se detecta un evento se agrega a la lista de tareas en espera, la cual se encuentra ordenada ascendentemente en función del tiempo límite de cada tarea.
- Entiéndase por tiempo límite: el máximo retardo que puede tener una tarea para ejecutarse antes de considerarse como fallida.

Planificación

Planificación dinámica

Algoritmo de mínima laxitud

- Laxitud (relajación) es la diferencia entre el tiempo en que debe ejecutarse una tarea y el tiempo en que puede ejecutarse realmente.
- Se mantiene una lista en orden ascendente en función de las laxitudes de cada tarea y se escoge en cada iteración la primera de la lista.

Sistemas distribuidos de manipulación de archivos

Sistemas de Operación II

Prof. Rodolfo Campos
rcampos@ucab.edu.ve

Universidad Católica Andrés Bello
Caracas, 2006

Archivos

- En muchos sistemas, como UNIX o DOS, un archivo es una secuencia de bytes donde su significado y estructura queda a cargo de las aplicaciones, sin embargo, muchos mainframes manejan varios tipos de archivos, cada uno con distintas propiedades.
- Los archivos tienen partes de información relativas (atributos), pero que no forman parte de ellos.

Sistemas distribuidos de archivos

"La tarea del sistema distribuido de archivos es almacenar los programas y datos, y tenerlos disponibles cuando sea necesario" – Tanenbaum, 1996

Servicio y Servidor de archivos

•Servicio de archivos. Representa la especificación de los servicios que el sistema de archivos ofrece a sus clientes. Describe las primitivas disponibles, los parámetros que utilizan y las acciones que llevan a cabo.

•Servidor de archivos. Se ejecuta en alguna máquina con el fin de ayudar a implantar el servicio de archivos.

Un sistema puede tener uno o varios servidores de archivos, pero los clientes no deberían conocer el número de servidores de archivos, su posición o función.

Sistemas distribuido de archivos

Componentes

- Servicio de archivos. Se encarga de las operaciones en los archivos individuales, como la lectura, escritura y adición.

- Servicio de directorios. Se encarga de crear y eliminar directorios, nombrar o cambiar el nombre de archivos y mover éstos de un directorio a otro.

Interfaz del servicio de archivos

- El servicio de archivos debe proveer primitivas para el manejo de las propiedades de los archivos.
- En algunos sistemas no se permite modificar archivos una vez creados, sólo se ofrecen las primitivas CREATE y READ. Los archivos en este tipo de sistemas reciben el nombre de inmutables.
- La protección de archivos generalmente se lleva a cabo mediante los esquemas tradicionales de posibilidades y lista de control de acceso.

Interfaz del servicio de archivos

Tipos de servicios de archivos

- Carga/descarga. En este modelo los archivos se trabajan localmente y luego se cargan en el servidor de archivos. En caso de requerir modificaciones a un archivo ya existente, se descarga del servidor de archivos, se modifica y se carga de nuevo.

- Acceso remoto. En este modelo los archivos se crean y modifican directamente en el servidor de archivos.

Interfaz del servicio de directorios

•La mayoría de los sistemas distribuidos permiten que los directorios contengan subdirectorios, para que los usuarios puedan agrupar los archivos relacionados entre sí.

•Los subdirectorios pueden contener sus propios subdirectorios y así en los sucesivo, lo que conduce a un árbol de directorios, el cual se conoce como sistema jerárquico de archivos.

•En ciertos sistemas se permite crear enlaces o apuntadores a directorios (ver comando *ln* en Linux), lo que permite crear no sólo árboles, sino gráficas arbitrarías de directorios. Nota: también se pueden enlazar archivos.

•Es importante conocer si todas las máquinas tienen la misma visión de la jerarquía de directorios.

Interfaz del servicio de directorios

Transparencia de los nombres

- Transparencia con respecto a la posición. Cuando el nombre de la ruta de acceso no sugiere la posición del archivo o directorio.
- Independencia con respecto a la posición. Cuando se pueden desplazar los archivos y directorios sin que cambien sus nombres en el sistema.

Interfaz del servicio de directorios

Métodos usuales para el nombramiento de archivos

- Nombre de máquina + ruta de acceso.
- Montaje de sistemas de archivos
- Un espacio de nombres que tenga la misma apariencia en todas las máquinas.

Interfaz del servicio de directorios

Ejercicio

Identifique el método y el nivel de transparencia en los siguientes casos:

- 192.168.0.1:/home/user
- /servidor1/user

Interfaz del servicio de directorios

Ejercicio

¿Proponga un método independiente con respecto a la posición?

Interfaz del servicio de directorios

Nombre de dos niveles

- Nombres simbólicos:
  - Son colocados por los usuarios.
  - Son cadenas ASCII, de un número máximo de caracteres y en algunos casos tienen extensiones.
- Nombres binarios:
  - Son de uso interno del sistema.
  - En UNIX pueden ser combinaciones de la máquina con el número nodo-i del archivo.

Métodos para compartir archivos

- Semántica de UNIX. El sistema impone un orden absoluto con respecto al tiempo y siempre devuelve el valor más reciente.
- Semántica de sesión. Los cambios realizados a un archivo sólo son visibles al resto, cuando éste es cerrado.
- Archivos inmutables. Los archivos no pueden ser modificados una vez creados.
- Transacciones. Todos los cambios cumplen la regla ACID.

Implantación

Características observadas de los sistemas de archivos

- La mayoría de los archivos son pequeños.
- La lectura es más común que la escritura.
- La lectura y la escritura son secuenciales generalmente.
- La mayoría de los archivos tienen una corta vida.
- Es poco usual compartir archivos.
- La mayoría de los procesos utilizan pocos archivos.
- Existen diferentes clases de archivos con propiedades diferentes.

Implantación

Estructura del sistema

- Puede existir diferencia entre un cliente y un servidor o no.
- Pueden combinarse el servicio de archivos y directorios en un servidor o pueden separarse en varios.
- Cuando se utiliza ocultamiento de nombres, es necesario informar al cliente de los cambios ocurridos.
- Pueden implementarse servidores que mantengan el estado conversacional con el cliente o no.

Implantación

Ventajas servidores sin estado y con estado

Sin estado	Con estado
Tolerancia a fallas	Mensajes de solicitud más cortos
No necesita llamadas OPEN/CLOSE	Mejor desempeño
No se desperdicia espacio en el Servidor en tablas	Es posible la lectura adelantada
No existe límite para el número de archivos abiertos	Es más fácil garantizar la idempotencia
No hay problemas si un cliente falla	Es posible la cerradura de archivos

Implantación

Ocultamiento

- En el disco del servidor.
- En la memoria principal del servidor.
- En el disco del cliente.
- En la memoria principal del cliente.
  - Dentro del propio espacio de direcciones de un proceso usuario.
  - Dentro del núcleo.
  - Dentro del espacio de direcciones de un proceso usuario administrador del caché.

Implantación

Métodos para administrar el caché de archivos del cliente

- Escritura a través del caché. Cuando se modifica una entrada del caché se envía de inmediato al servidor.
- Escritura retrasada. Se envían al servidor bloques de modificaciones realizadas sobre el caché.
- Escritura al cierre. Se envían los cambios realizados sobre el caché sólo cuando se cierra el archivo; se apega a la semántica de sesión.
- Control centralizado. Los cambios realizados sobre un archivo son enviados a todos los caché de clientes por el servidor; se apega a la semántica de UNIX.

Implantación

Réplica

- Réplica explícita de archivos. El cliente debe copiar los archivos por su cuenta en todos los servidores.
- Replica retrasada de archivos. El cliente copia los archivos en un servidor primario y éste luego los replica en servidores secundarios.
- Replica de archivos mediante grupo. El cliente copia los archivos al grupo y este se encarga de distribuirlos.

Un sistema es transparente con respecto a réplicas cuando los usuarios no necesitan manejar el proceso de réplica.

Implantación

Actualización mediante el voto

- Para actualizar un archivo replicado en N servidores, el cliente debe establecer contacto con al menos, la mitad más uno de los servidores (la mayoría), y ponerlos de acuerdo para realizar la actualización.
- Para leer un archivo el cliente debe pedir a la mayoría, que le envía la versión de sus archivos y en caso de que éstas concuerden, él puede estar seguro de que la lectura que desea realizar es confiable.

Servicios Web

Sistemas de Operación II

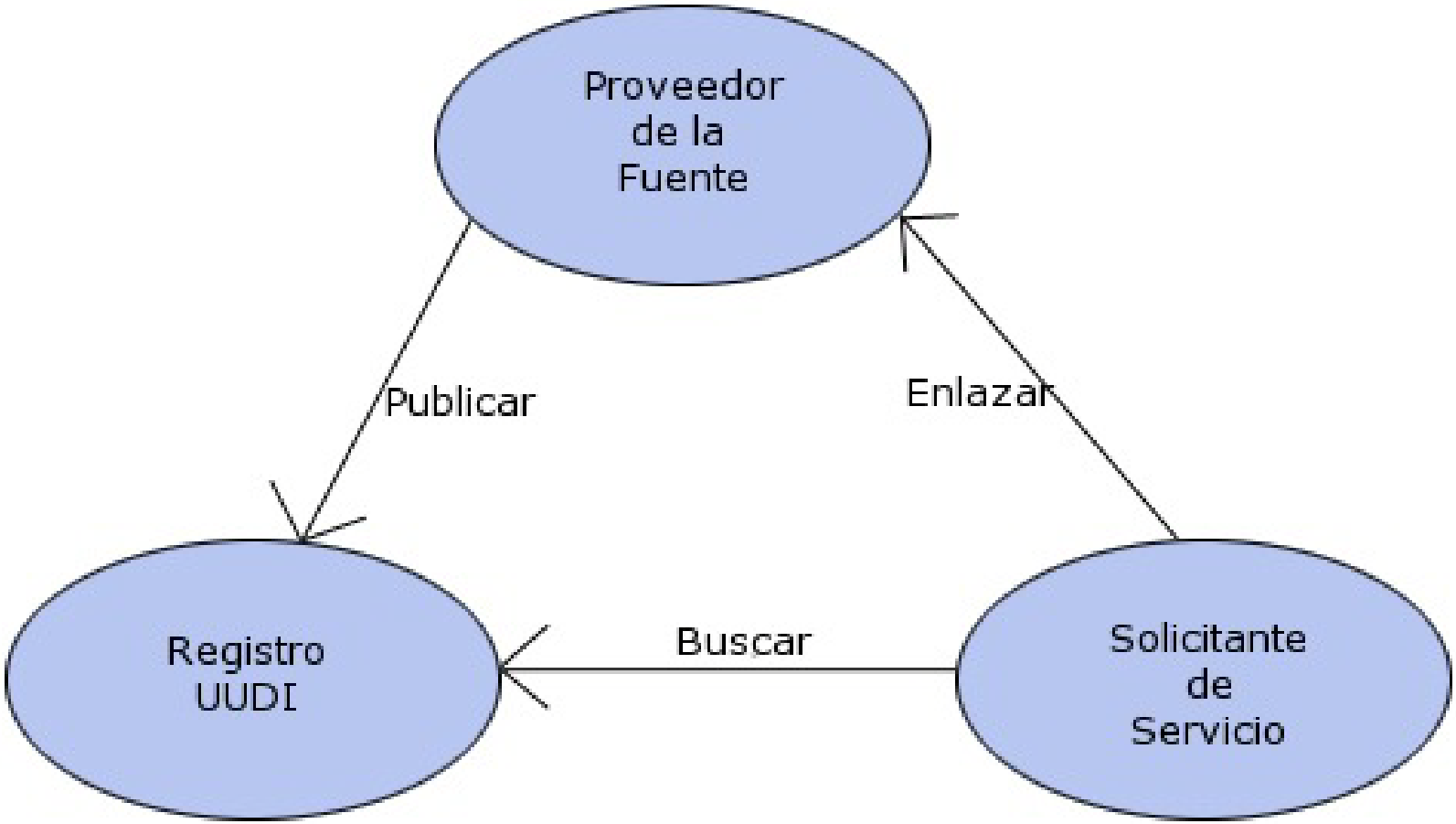
Prof. Rodolfo Campos
rcampos@ucab.edu.ve

Universidad Católica Andrés Bello
Caracas, 2006

Servicios Web

- Un Servicio Web es una tecnología que permite al usuario acceder a un URL específico usando protocolos estándar para recibir servicios.
- Los Servicios Web combinan el poder de las tecnologías:
  - XML
  - HTTP
- Los servicios Web son auto contenidos, auto descriptivos y modulares.
- Los servicios Web pueden ser publicados, localizados e invocados a través de la Internet.

Arquitectura



Tecnologías involucradas

- HTTP forma la base de los Servicios Web.
- Un Servicio Web es llamado a través de un Protocolo de Internet estándar.
- XML es la opción natural para formatear los datos.
- SOAP es un protocolo basado en XML, el cual se usa para ejecutar RPC sobre HTTP.

Características

- Auto contenidos.
- Basados en estándares abiertos y tecnologías.
- Pueden desempeñar un amplio rango de funciones.
- Débilmente acoplados.
- Servicios entregados en base a demanda

Acerca de la Implementación

Los Servicios Web deben ser:

- Creados.
- Descritos.
- Publicados para que los usuarios los localicen.
- Localizables por los usuarios potenciales.
- Invocados.
- Retirados cuando no se requieren más

Acerca del Desarrollo

- La mayoría de Servicios Web usan lenguajes de programación OO como Visual Basic .NET o Java.
- Una vez que la aplicación se desarrolla, un descriptor se crea para describir la aplicación como un servicio Web. Éste debe ser escrito utilizando el Lenguaje Descriptor de Servicios Web (WSDL).
- El archivo WSDL que se crea puede ser registrado con el registro UDDI, que se usa para buscar los métodos.
- Las peticiones y respuesta viajan a través de la Internet en forma de:
    - SOAP
    - HTTP GET / POST
    - MIME (Multi-Porpuse Mail Extensions)

WSDL

- La interfaz WSDL es un documento XML, que contiene información relevante acerca de los Servicios Web. Ver: http://www.w3.org/TR/wsdl
- La ubicación donde reside la aplicación está también presente en la interfaz WSDL.
- La interfaz WSDL la usa el solicitante del servicio para acceder a cualquier servicio.
- El agente intermediario usa esta interfaz para buscar un servicio requerido.
- El documento WSDL tiene dos secciones:
  - **Interfaz:** describe lo que proporciona el servicio y su protocolo.
  - **Implementación:** describe la información de ubicación del acceso al servicio.

SOAP

- Los mensajes SOAP (Simple Object Access Protocol) son documentos XML. Ver: http://www.w3.org/TR/soap
- La información del pedido y respuesta se almacenan como una parte de XML.
- Los elementos que forman los mensajes SOAP son los siguientes:
  - **Sobre o Envoltura SOAP:** Define el contenido de mensaje.
  - **Encabezado SOAP:** Es opcional y tiene una información de encabezado.
  - **Cuerpo SOAP:** Tiene información de llamada y respuesta.

Tópicos de rendimiento

- Los protocolos de mensajería como CORBA, DCOM, y RMI usan codificación binaria para argumentos y valores de retorno.
- Toda la comunicación sucede bajo la suposición de que ambos, el emisor y el receptor conocen plenamente el presente texto.
- SOAP usa XML para codificar mensajes, haciendo el procesamiento de mensajes una tarea fácil.
- SOAP se usa para enviar mensajes a través de la Internet.
- El tiempo necesario para codificar y decodificar los mensajes en cada punto final es menor comparado con el tiempo que toma transferir los bytes entre puntos finales