Sistemas Distribuidos

Tema 2: EL MODELO CLIENTE/SERVIDOR

E. U. Informática en Segovia
Departamento de Informática
Universidad de Valladolid

Sistemas Distribuidos

Definición de sistemas cliente/servidor (1)

- Clientes y servidores: entidades lógicas separadas por una red para lograr cierta tarea. Diferencias con otro software distribuido:
- Servicio: relación cliente/servidor entre procesos que se ejecutan en máquinas distintas diferenciados en función de la idea de servicio:
  - Servidor: proveedor de servicios
  - Cliente: consumidor de servicios
- Recursos compartidos: servidor puede dar servicio a más de un cliente y regular su acceso a recursos compartidos
- Asimetría de protocolos:
  - Relación de muchos a uno
  - El cliente comienza el diálogo solicitando servicio; el servidor espera solicitudes
- Localización transparente: el software cliente/servidor enmascara la localización del servidor
- Independencia del hardware

Definición de sistemas cliente/servidor (2)

Más características:

- Intercambio de mensajes: clientes y los servidores son procesos débilmente acoplados que interaccionan mediante paso de mensajes
- Encapsulación de servicios: posible actualización de servidores si no se modifica el protocolo
- Escalabilidad: relación cliente/servidor entre procesos que se ejecutan en máquinas distintas diferenciados en función de la idea de servicio:
  - Horizontal: añadir clientes sin que el servicio se vea demasiado afectado
  - Vertical: se puede cambiar el servidor o distribuir el servicio por múltiples servidores
- Integridad: datos centralizados en el servidor => mantenimiento de integridad sencillo

Tecnología cliente/servidor(1)

- Servidores de ficheros: los clientes hacen solicitudes de ficheros al servidor: forma de compartir ficheros en una red (repositorios de documentos, imágenes, programas, etc.)
- Servidores de bases de datos: aplicaciones del cliente mandan solicitudes SQL al servidor. El servidor devuelve el resultado de la consulta.
- Servidores de transacciones: el cliente invoca procedimientos remotos o transacciones (conjunto de instrucciones SQL) sobre la base de datos. Los datos intercambiados son:
  - Cliente -> servidor: solicitud
  - Servidor -> cliente: mensaje de resultado
- Servidores groupware: intercambio de información semiestructurada: texto, imágenes, u otros (Lotus Notes o Microsoft Exchange). Cada vez más se usa e-mail

Tecnología cliente/servidor(2)

- Servidores de aplicaciones de objetos:
  Aplicación cliente/servidor: conjunto de objetos de comunicación. Los objetos del cliente usan un Object Request Broker (ORB). El cliente invoca un método remoto, el ORB localiza una instancia de la clase del objeto en el servidor, invoca el método y devuelve el resultado al objeto del cliente.
  CORBA (Common Object Request Broker Arquitecture)

- Servidores de aplicaciones web:
  World Wide Web: arquitectura cliente/servidor (los clientes solicitan documentos a los servidores). La solicitud es por nombre y el protocolo es HTTP
  Hay objetos web y toda clase de aplicaciones nuevas

Modelo de cliente grueso/fino

- Modelo de cliente grueso: funcionalidad en el cliente. Servidores de ficheros o de bases de datos, (el cliente conoce la organización de los datos en el servidor)

- Modelo de servidor grueso: funcionalidad en el servidor. Minimiza la carga del cliente y de la red. En el cliente interfaz de usuario que interacciona con el servidor mediante llamadas remotas

  A menudo complementarios: aplicación con servicios de ficheros, base de datos, transacciones y objetos.

Arquitecturas cliente/servidor de 2, 3 ó N niveles (1)

Arquitecturas 2-tier, 3-tier y N-tier
Clasificación lógica

- 2 niveles: lo visto hasta ahora. Ej: servidor de páginas web

  Cliente (navegador) <-> http <-> Servidor (documentos)

- 3 niveles: parte de la lógica de la aplicación en un nivel intermedio. Ej: acceso a base de datos a través de web
  - Cliente web : petición
  - Servidor web: recoge consulta y es cliente del servidor de base de datos
  - Servidor de base de datos

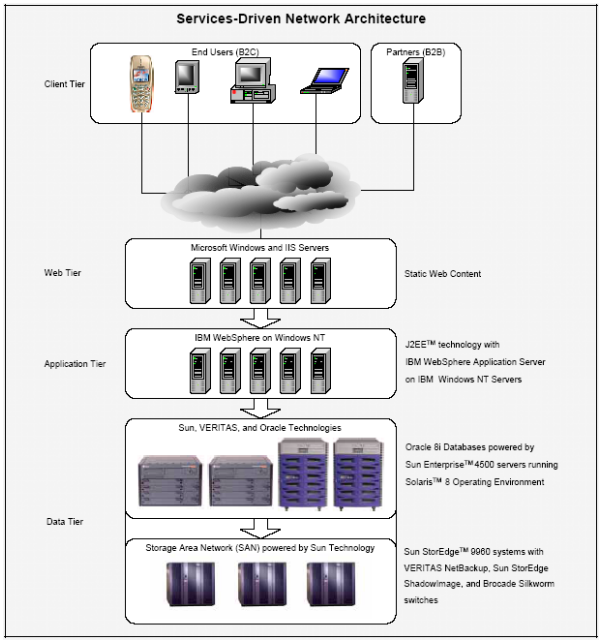
  Ventajas principales:
  - Encapsulación => flexibilidad, escalabilidad…
  - Seguridad

Arquitecturas cliente/servidor de 2, 3 ó N niveles (2)

- N niveles: descomposición del nivel intermedio en un grupo de componentes que interactúan. Ej: portales de compra en Internet

Ventajas principales:

- Desarrollo más fácil y flexible
- Reutilización de componentes
- Encapsulamiento => aplicaciones más seguras
- Mejora continua del servicio



Configuraciones cliente/servidor típicas

Tres bloques básicos en una arquitectura cliente/servidor:
cliente, servidor, middleware

Ejemplos de configuraciones:

- Arquitecturas cliente/servidor en la misma máquina. Ej: sistema de gestión de una consulta de un médico con un solo ordenador => alta escalabilidad
- Arquitecturas cliente/servidor con servidor único. Ej: sistemas basados en LAN con un servidor y varios clientes (terminales)
- Arquitecturas cliente/servidor con varios servidores.
  - Varios servidores con funciones distintas
  - Duplicación de servidores para robustez frente a fallos o para aumento de rendimiento más fácil y flexible
- Arquitecturas cliente/servidor en la cual cada máquina que es un cliente y un servidor completo

Elementos de arquitecturas cliente/servidor

- Tres bloques básicos en una arquitectura cliente/servidor:
  - Cliente: incluye sistema operativo (OS) sobre con interfaz gráfico de usuario (GUI) o interfaz orientado a objetos de usuario (OOUI)
  - Servidor: ejecuta software especializado
  - Middleware: software distribuido para interacciones entre cliente y servidor
    Desde la API del cliente usada para invocar el servicio, la transmisión de la solicitud y la respuesta hasta el sistema que informa al servidor
    No incluye el software que proporciona el servicio ni el interfaz de usuario en el cliente. Parte en el cliente y parte en el servidor
    El middleware incluye:
    - Protocolos de transporte, como TCP/IP, IPX...
    - NOS's (Sistemas operativos de red), como RPC, Samba...
    - Middleware específico para el servicio como HTTP, ORB...
    - Responsable del buen funcionamiento, especialmente en N niveles
- DSM (*Distributed System Management*): agentes que envían información desde todos los nodos de la red cliente/servidor para su recogida y presentación

Características del servidor: funciones del servidor

- Esperar peticiones de clientes (mensajes). A veces sesión por cliente y otras conjunto dinámico de sesiones
- Atender solicitudes simultáneas => concurrencia. Sin riesgo para la integridad de los recursos compartidos
- Prioridades en la atención de las solicitudes
- Capacidad de lanzar tareas en segundo plano no relacionadas con el servicio Ejemplo: un servidor de ftp aprovecha las horas de la noche para actualizar un mirror
- Robustez: crítica en servidores
- Escalabilidad y extensibilidad

Características del servidor: requisitos del SO del servidor (1)

Distinguimos en un SO servicios básicos ("de serie") y servicios extendidos:

SERVICIOS BÁSICOS:

- Alto nivel de concurrencia (tanto de tareas como en cada tarea)
- *Task preemption*: fin de las tareas "voluntario" => peligro. Mejor slots de tamaño fijo
- Prioridades
- Mecanismos de concurrencia (semáforos, monitores)
- Mecanismos de comunicación entre procesos. Redireccionamiento transparente
- Threads
- Sistema de ficheros multiusuario de altas prestaciones: muchos ficheros abiertos simultáneamente y protección de integridad
- Sistema eficaz de gestión de memoria: manipulación de objetos y programas grandes. Sistema de intercambio ágil con el disco
- Extensibilidad sin recompilar o (idealmente) rearrancar

Características del servidor: requisitos del SO del servidor (2)

SERVICIOS EXTENDIDOS:

- Soporte para distintos protocolos de comunicación => servicio a clientes distintos
- Extensiones para acceso transparente a recursos compartidos (ficheros, impresoras...)
- Recursos de manipulación de BLOBs (Binary Large OBjects): imágenes, video, gráficos...
- Sistema de directorio global o páginas amarillas (localización de recursos por su nombre)
- Servicios de autenticación (un cliente es quien dice ser) y autorización (un cliente puede hacer lo que está haciendo)
- Gestión del sistema: configuración, monitorización, generación de alertas, distribución y manipulación de paquetes de software para los clientes, identificación de virus o intrusos...
- Sincronización temporal entre clientes y servidor
- Servicios de bases de datos y de transacciones
- Servicios de internet: HTTP, SSL, firewalls, DNS...
- Servicios orientados a objetos

Características del servidor: evolución de los servidores (1)

Segmento muy heterogéneo:

- desde servidores simples de impresora
- hasta servidores de clusters para procesamiento masivo

El segmento más importante (comercialmente): servidores de aplicaciones (web, bases de datos, objetos, groupware...)

Por segmentos:

- Segmento bajo y medio: NetWare, Microsoft y Unixes (Solaris, Linux, FreeBSD)
- Segmento alto: casi exclusivamente Unix

Características del servidor: evolución de los servidores (2)

CARACTERÍSTICAS:

- NetWare:
  - Buen servidor de ficheros; mal servidor de aplicaciones
  - Soporta clientes de Windows, Mac y Linux
  - Incorpora LDAP, una máquina virtual Java, CORBA, etc.
- Microsoft:
  - Servidor de aplicaciones, de ficheros e impresora y de bases de datos
  - Bien con clientes Windows e incorpora herramientas de Microsoft
  - Problemas de Microsoft y no es fácilmente escalable
  - Mal para multiprocesador
- Unix:
  - A bajo nivel, linux+apache predomina en Internet
  - A nivel medio, Unix y Linux
  - A alto nivel Unix

Características del cliente

Dividimos los clientes en tres tipos:

- Sin GUI (Interfaz gráfico de usuario):
  - Lectores de códigos de barras, demonios...
- Con GUI:
  - Sustituyeron a los terminales sin gráficos
  - Normalmente usan el modelo objeto/acción: selección de objetos y acciones para realizar sobre éstos.
  - Normalmente los diálogos de naturaleza secuencial
  - Ejemplos: SOs antiguos o las páginas web con formularios
- Con OOUI (Interfaz de usuario orientada a objetos):
  - El usuario manipula de forma objetos en pantalla (*drag-and-drop*)
- Diferencias entre GUI y OOUI:
  - Los OOUI en realidad extensiones del interfaz del sistema operativo => no es fácil decir donde acaba la aplicación y empieza el SO
  - GUI: icono=aplicación
  - OOUI: icono=objeto
- Ejemplos:
  - GUI: Windows 3.X, Motif y páginas web sencillas
  - OOUI: MacOs, Windows 32 bits, Gnome, KDE y páginas web que utilizan Java 2 JavaBeans.

Características del cliente: requisitos del SO del cliente

- Todos necesitan mecanismo para implementar el mecanismo de solicitud/respuesta (evidente)
- Todos necesitan algún tipo de transferencia de ficheros (intercambio de imágenes, texto...)
- Facilidades multitarea (prioridades, preasignación temporal de tareas, comunicación entre procesos, threads)
  - Imprescindible en clientes sin GUI con multitarea en el servicio y clientes con OOUI
  - Para GUI simples viene bien
  - Para clientes sin GUI y sin multitarea no es necesario
- Portabilidad de código: máquina virtual Java en los clientes
- Robustez: el servidor no controla los clientes => evitar que un proceso de cliente dé problemas en el servicio

Características del cliente: evolución de los clientes

Sector en cambio vertiginoso

Tendencias en la evolución de los clientes:

- SO de los clientes cada vez menos monopolizado: hace años sólo Windows 3.X y el DOS. Ahora, Windows distintos, MacOS X, Linux, PalmOs...
  - Cliente universal: navegador de internet
  - Diversificación de PCs: PC's supergruesos y PC's superfinos
  - Cada vez más clientes incrustados en dispositivos portátiles

SOs más habituales en los clientes:

- Windows:
  - Ventajas: interfaz muy conocido; soporta distintos protocolos como TCP/IP, NPX/SPX, PPP...
  - Desventajas: coste, seguridad y acaparador de recursos
- Mac OS X:

  Presencia en Internet muy superior a su presencia en ordenadores personales
  - Ventajas: entorno gráfico magnífico; basado en FreeBSD
  - Desventajas: coste, software, y mal soporte a hardware de otros fabricantes
- Linux:
  - Ventajas: fiabilidad, prestaciones, gratuidad, soporte sobre distintos sistemas
  - Desventajas: instalación, Office, un buen emulador de windows gratuito.

El middleware: objetivos del middleware

Función del middleware: que todo funcione con trasparencia
El sistema da la impresión de ser único
Internet = sistema único de millones de usuarios

Tipos de transparencia:

- De localización: innecesario saber la localización de un recurso \\Máquina\directorio\fichero viola la transparencia de localización
- De nombres: mismo espacio de nombres en toda la red
- De acceso: sistema de acceso único
- De replicación: trabajar con recursos duplicados como únicos. Ej: mantener las copias y sincronizar una base de datos replicada
- De acceso local o remoto: acceder a distancia igual que localmente
- Temporal: mantenimiento de relojes de todo el sistema
- De fallos: el NOS debe controlar reintentos y recuperaciones
- De administración: interfaz de administración única y consistente

El middleware: funcionalidades del middleware

El middleware debe ofrecer las siguientes funcionalidades:

- Sistema de ficheros distribuido
- Servicio de directorio global
- Servicio de tiempo distribuido
- Mecanismos de seguridad
- Sistemas de comunicaciones a través de la red:
  - Un sistema de comunicaciones punto a punto
  - Un sistema de invocación remota de procedimientos
  - Un sistema de mensajería de red