Sistemas Operativos Distribuidos

Introducción

Contenido del tema

- Definición de sistema distribuido.
- Ventajas y desventajas de los sistemas distribuidos.
- Modelos de computación distribuida.
- Objetivos de un sistema distribuido.
- Arquitectura software de los sistemas distribuidos.
  - Sistemas operativos distribuidos versus *Middlewares*.
- Componentes de un sistema distribuido.

Sistema distribuido (SD)

- Conjunto de procesadores conectados por una red:
  - Sin memoria compartida
    - Sistema débilmente acoplado
  - No existe un reloj común
  - Dispositivos de E/S asociados a cada procesador
  - Fallos independientes de componentes del SD
  - Carácter heterogéneo
- Objetivo de la asignatura: Software de sistema del SD
  - Sistemas Operativos Distribuidos
  - Interfaz software que oculta la complejidad hardware de un SD
    - Idealmente, visión de sistema único (*Single System Image*)
- Término afín: Computación distribuida
  - Ejecución de una aplicación en un SD

Ventajas de los Sistemas Distribuidos

- Economía: Buena relación rendimiento/coste
  - Avances en tecnología de microprocesadores y redes de área local.
- Alto rendimiento: Procesamiento paralelo.
- Soporte de aplicaciones inherentemente distribuidas.
  - Por ejemplo: empresa distribuida geográficamente
- Capacidad de crecimiento: Escalabilidad.
- Fiabilidad y disponibilidad: Tolerancia a fallos.
- Carácter abierto y heterogéneo:
  - Estándares de interoperabilidad.
- Compartir recursos y datos.

Desventajas de los Sistemas Distribuidos

- Necesidad de un nuevo tipo de software:
  - Más complejo.
  - No hay todavía un acuerdo sobre cómo debe ser.
- Red de interconexión introduce nuevos problemas:
  - Pérdida de mensajes y saturación.
  - Latencia puede provocar que al recibir un dato ya esté obsoleto.
  - La red es un elemento crítico.
- Seguridad y confidencialidad
- Definición alternativa de SD:
  - "*Un sistema distribuido es aquél en el que no puedes trabajar con tu máquina por el fallo de otra máquina que ni siquiera sabías que existía*" (Lamport)

Fallacies of Distributed Computing

- *The network is reliable.*
- *Latency is zero.*
- *Bandwidth is infinite.*
- *The network is secure.*
- *Topology doesn't change.*
- *There is one administrator.*
- *Transport cost is zero.*
- *The network is homogeneous.*

- 7 primeras propuestas en 1994 por Peter Deusch (Sun)
  - Octava por James Gosling (Java/Sun) en 1996

Aplicaciones de los Sistemas Distribuidos

- Entornos empresariales: redes corporativas e *intranets*:
  - Sustituye a los clásicos *mainframes*.
  - "Sistema de información distribuido"
- Entornos de computación de altas prestaciones:
  - Procesamiento paralelo, alternativa a costosos *supercomputadores*.
  - "Sistema de computación distribuido"
- Servicios con alta disponibilidad y rendimiento.
- Sistemas distribuidos de gestión de bases de datos
- Aplicaciones multimedia.
- Sistemas industriales distribuidos y aplicaciones de control.
- Internet: un enorme sistema distribuido.
- Ubicuos: automóviles, electrodomésticos, edificios, ...

Modelos de computación distribuida

- *Cluster Computing*
- *Utility Computing*
- *Grid Computing*
- *Volunteer Computing*
- *Cloud Computing*
- *Autonomic Computing*
- *Mobile (Nomadic) Computing*
- *Ubiquitous (Pervasive) Computing*

Definiciones no excluyentes y, en algunos casos, sin acuerdo general

Cluster Computing

- SD dedicado a ejecutar una aplicación buscando
  - Altas prestaciones y/o alta disponibilidad.
  - Puede servir varias aplicaciones mediante partición
- Alternativa a supercomputadores con mejor calidad/precio
- Características usuales:
  - Más fuertemente acoplado que SD general
  - Poca dispersión geográfica
  - Redes de alta velocidad
  - Sistema homogéneos
    - Aunque puede usar sistemas heterogéneos y virtualización
  - Carácter estático
  - Uso habitual de componentes hardware estándar
- Aspectos software más relevantes:
  - Entorno para desarrollo de aplicaciones paralelas
  - Planificación de trabajos

Utility Computing

- Computación como otra empresa de servicio público
  - "Alquiler" de recursos computacionales externos
  - Demanda dinámica basada en necesidades puntuales
    - *On-demand Computing*
- Define un modelo de trabajo más que una plataforma
  - Aunque la solución natural es algún tipo de SD
    - Sistemas de computación *grid* o *cloud*
- No es una idea nueva: John McCarthy (1961)
  - *"Computing may someday be organized as a public utility just as the telephone system is a public utility."*
- Uso habitual de virtualización
- Problema de la tarificación

Grid Computing

- Acuñado por Foster inspirado en *power grid*
- Extensión de *cluster computing* a mayor escala:
  - Máquinas con mayor dispersión geográfica
  - Menor grado de acoplamiento
  - Pueden extenderse a varios dominios de administración
    - Desde interdepartamentales hasta intercorporativos
  - *Grid* = "*Cluster* virtual" sobre máquinas de varias organizaciones
    - Recursos no dedicados
    - *Grid* convive con SD de cada organización
  - Sistemas heterogéneos (virtualización)
  - Sistemas dinámicos
- Aspectos software relevantes:
  - Coordinar recursos de varias organizaciones sin un control central
  - Uso de estándares y sistemas abiertos
  - Seguridad

Volunteer Computing

- SD formado por recursos donados por usuarios a proyectos
  - Normalmente, ciclos de procesador y espacio de almacenamiento
  - Carácter altruista (Folding@home, SETI@home)
- Similar a computación *grid*
  - Dinámico, recursos no dedicados, dispersión geográfica, ...
- Pero con diferencias:
  - Implica individuos, no organizaciones
  - Asimetría de roles: usuario-proyecto
    - Simetría del *grid*: organización-organización
  - Mayores problema de seguridad
    - Usuarios anónimos

Cloud Computing

- Recursos HW y/o SW ofrecidos como servicio (¿de pago?)
  - Recursos virtualizados y dinámicamente escalables
  - Siguiente etapa en la evolución *grid-utility* basada en Internet
    - *Cloud* metáfora frecuente de Internet
- Modelos de uso:
  - *Infrastructure as a Service* (ej *Amazon Elastic Compute Clouds*, EC2)
    - Oferta dinámica de recursos HW virtuales según necesite cliente
  - *Platform as a Service* (ej. *Google App Engine*)
    - Además plataforma SW de desarrollo
  - *Software as a Service* (ej. *Google Apps*)
    - Además aplicaciones de interés
- ¿Futura convergencia *cloud* y *grid*?
- Debate sobre aspectos sociales
  - ¿Un paso más hacia la globalización? ¿Mayor pérdida de privacidad?

Autonomic Computing

- Sistemas cada vez más complejos: necesidad de autogestión
  - Iniciativa de IBM aplicable especialmente a SD
    - Inspirado por sistema nervioso autónomo
- 4 áreas funcionales:
  - Auto-configuración
  - Auto-reparación
  - Auto-optimización
  - Auto-protección
- 5 niveles evolutivos de implantación:
  - Desde gestión manual de componentes aislados
  - Hasta gestión automatizada del sistema en su integridad
    - Uso de bucle de control cerrado (con realimentación)

Mobile Computing

- SD incluye dispositivos portátiles con acceso remoto
  - Proliferación de dispositivos portátiles y redes inalámbricas
  - Usuario accede a su organización mientras viaja
- Aspectos a considerar
  - Limitaciones en los recursos del dispositivo
  - Control de consumo de energía del dispositivo
  - Ancho de banda variable
  - Modo desconectado
    - Usuario debe poder trabajar sin conexión
    - Al reconectarse, “reconciliación” entre info. en dispositivo y en SD
      - Puede ser automática o manual
  - Mayores amenazas a la seguridad y privacidad

Ubiquitous Computing

- Computadores omnipresentes incluidos en todo tipo de objetos
- SD formado por los dispositivos de cómputo en un ámbito
- Aspectos a considerar
  - Sistema dinámico: componentes (des)aparecen y se mueven
  - *Spontaneous Networking*
  - Localización de dispositivos/usuarios
  - *Wearable Computing*
  - *Context-aware Computing*
- Escenarios de ejemplo
  - Imprimir fotos al llegar a un hotel
  - Visita guiada a un museo

Objetivos de un Sistema Distribuido

- Transparencia
- Rendimiento
- Capacidad de crecimiento
- Carácter abierto
- Fiabilidad

Transparencia

Existen varios perfiles de transparencia:

- **Acceso**: Manera de acceder a recurso local igual que a remoto.
- **Posición**: Se accede a los recursos sin conocer su localización.
- **Migración**: Recursos pueden migrar sin afectar a los usuarios.
- **Concurrencia**: Acceso concurrente no afecta a los usuarios.
- **Replicación**: La existencia de réplicas no afecta a los usuarios.
- **Fallos**: La ocurrencia de fallos no afecta a los usuarios.
- **Crecimiento**: El crecimiento del sistema no afecta a los usuarios.
- **Heterogeneidad**:Carácter heterogéneo no afecta a los usuarios.

- No siempre se puede conseguir
- Ni siempre es buena:
  - Diseñadores de Java RMI consideran que la invocación de métodos remota no debe ser exactamente igual que la local
    - "*A Note on Distributed Computing*", Jim Waldo, 1994

Rendimiento

Rendimiento para un **servicio multiusuario**:

- Objetivo: Rendimiento no peor que un sistema centralizado

Rendimiento para la **ejecución paralela** de aplicaciones:

- Objetivo: Rendimiento proporcional a procesadores empleados

Factores:

- Uso de esquemas de caché
  - Intentar que muchos accesos se hagan localmente
- Uso de esquemas de replicación
  - Reparto de carga entre componentes replicados
- En ambos casos: Coste de mantener la coherencia

Capacidad de crecimiento

Diseño de un sistema distribuido **debe evitar** "cuellos de botella":

- Componentes centralizados
- Tablas centralizadas
- Algoritmos centralizados

Estrategias:

- Reparto de estructuras de datos entre varios nodos.
- Replicación y caché
- Realización de parte del procesamiento en los nodos cliente.

Características deseables en un algoritmo distribuido:

- Ninguna máquina tiene información completa del estado del sistema
- Las decisiones se basan sólo en información disponible localmente
- El fallo de una máquina no debe invalidar el algoritmo
- No debe asumir la existencia de un reloj global

Carácter abierto

- SD abierto: servicios, protocolos, etc. publicados y estándares
- Facilita la interacción con otros sistemas abiertos
- Posibilita migración de aplicaciones a/desde otros SD abiertos
- Flexibilidad para cambiar y extender el SD
- Esconde heterogeneidad de HW, SO, lenguajes, ...

Fiabilidad

- Teóricamente: OR-lógico de sus componentes.
- Sin embargo, a veces: AND-lógico de varios componentes.
- Evitar componentes críticos (punto único de fallo).
- Uso de replicación activa o pasiva
  - Mantenimiento de coherencia entre réplicas
- Operación correcta en sistema particionado por error de red
  - “Reconciliación” al reintegrarse

Software de sistema de los SD

- SO para máquina con m. compartida no válido para SD
  - SW **fuertemente** acoplado sobre HW **fuertemente** acoplado
- Primeros sistemas: “Sistemas Operativos de Red”
  - Sistema operativo convencional + utilidades de red.
  - Protocolos de comunicación para acceso a recursos.
  - Cada máquina una copia de SO (posiblemente distinto).
  - SW **débilmente** acoplado sobre HW **débilmente** acoplado
- Software de sistema de SD debería hacer que:
  - usuarios lo perciban como sistema centralizado (*single system view*)
  - SW **fuertemente** acoplado sobre HW **débilmente** acoplado
- Dos arquitecturas software alternativas:
  - Sistemas Operativos Distribuidos
  - *Middlewares*

Sistemas Operativos Distribuidos (SOD)

- Una copia del mismo SO en cada procesador
- Necesidad de desarrollar nuevos conceptos
- Algunos ejemplos de esta problemática específica:
  - ¿Cómo lograr exclusión mutua sin memoria compartida?
  - ¿Cómo tratar los interbloqueos sin un estado global?
  - Planificación de procesos: Cada copia del sistema operativo tiene su cola de planificación (migración de procesos).
  - ¿Cómo crear un árbol de ficheros único?
  - Implicaciones de no reloj único, presencia de fallos o heterogeneidad.
- SOD revolución: ¿tiramos a la basura nuestros SSOO viejos?
- Mejor evolución: *middleware*.

Middleware

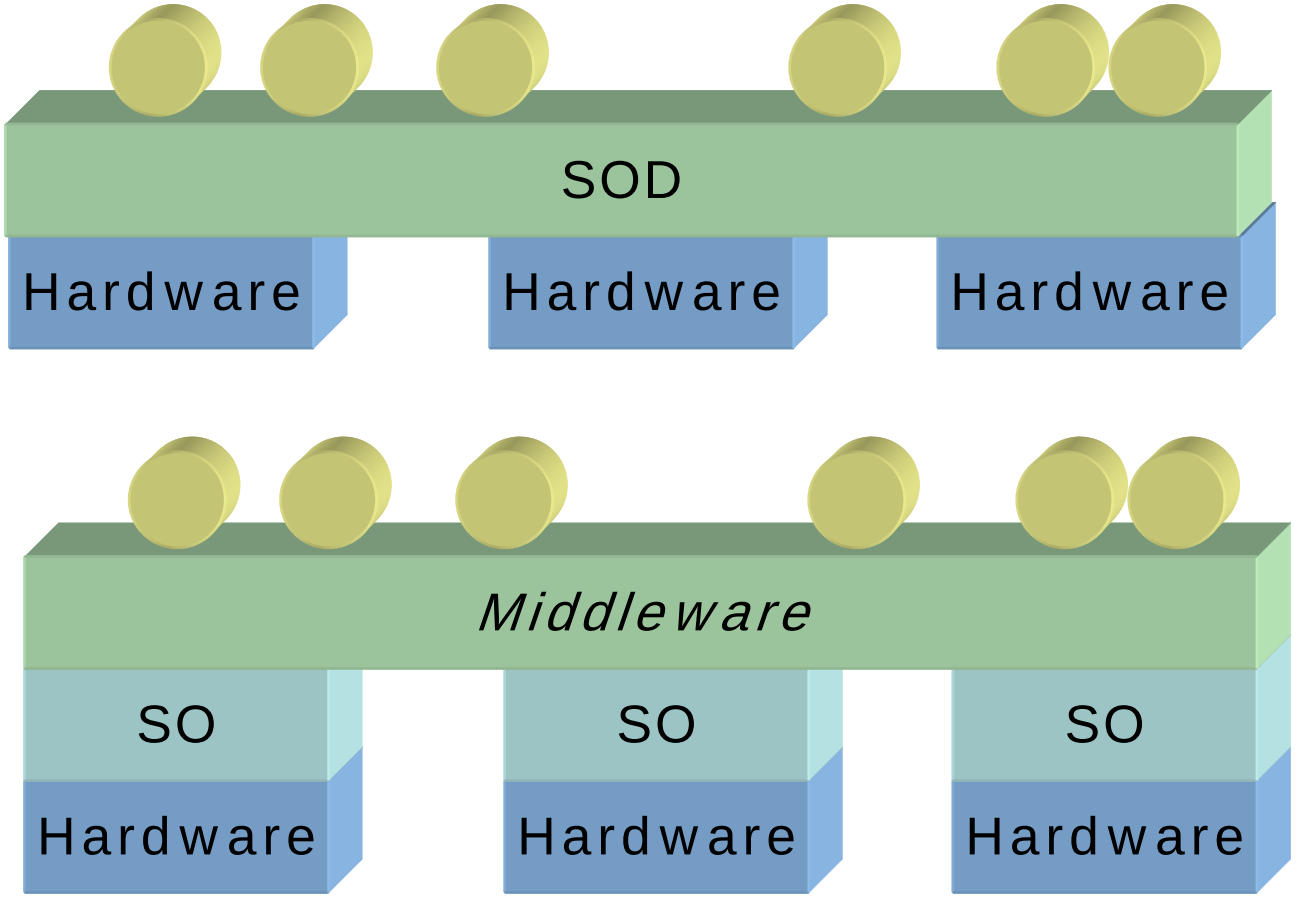
Middleware:

- Capa de software que ejecuta sobre el sistema operativo local ofreciendo unos servicios distribuidos estandarizados.
- Sistema abierto independiente del fabricante.
- No depende del hardware y sistema operativo subyacente.

Ejemplos:

- DCE (Open Group).
- CORBA (OMG).

SOD versus *Middleware*



Componentes de un Sistema Distribuido

Componentes = Temario

- Servicios de comunicación.
- Servicio de nombres.
- Sistemas de ficheros distribuidos.
- Gestión de procesos.
- Memoria compartida distribuida.
- Servicios de sincronización y coordinación.
- Servicio de tolerancia a fallos y seguridad.

Servicios de comunicación

- Arquitecturas de comunicación:
  - Cliente/servidor
  - Editor/subscriptor
  - *Peer-to-peer*
  - Arquitecturas para computación distribuida (p.e. maestro/trabajador)
- Tecnologías de comunicación:
  - Paso de mensajes (*sockets*)
  - Llamada a procedimientos remotos (RPC).
  - Invocación de métodos remotos (RMI).
  - Tecnologías de objetos distribuidos (CORBA).
  - Servicios web.
  - Arquitecturas orientadas a servicios (SOA).

Servicio de nombres

- Composición del espacio de nombres
- Distribución y replicación del espacio de nombres
- DNS
- Servicio de directorio
- LDAP
- Servicio de descubrimiento

Sistemas de Ficheros Distribuidos

- Estructura de un SFD
- Resolución de nombres
- Acceso a los datos
- Gestión de cache
- Gestión de cerrojos
- NFS, AFS y Coda
- Sistemas de ficheros paralelos

Gestión de procesos

- Caracterización de la carga:
  - Consumo de CPU.
  - Consumo de otros recursos (Memoria)
  - Prioridades.
- Estrategias de asignación de procesadores a procesos:
  - Cuándo, cuál y a dónde.
- Planificación de procesos:
  - Planificación interna.
  - Planificación global.
- Migración de procesos
  - Equilibrado de carga.
  - Aprovechamiento de máquinas inactivas.

Memoria Compartida Distribuida (DSM)

- Concepto:
  - Memoria físicamente privada pero lógicamente compartida.
- Estrategias de implementación:
  - Basada en páginas.
  - Basada en variables compartidas.
- Aspectos de diseño de DSM
- Modelos de coherencia
- DSM basada en espacios de tuplas

Sincronización y Coordinación

Comprende los conceptos de:

- Tiempo en entornos distribuidos: Sincronización de relojes y relojes lógicos.
- Concurrencia y Paralelismo: Exclusión mutua e interbloqueos.
- Algoritmos distribuidos: Elección de líder, consenso, ...
- Transacciones: Propiedades ACID, modelos de *commit/rollback*.

Afecta a otros servicios:

- Nombrado e identificación.
- Seguridad y fiabilidad.
- Comunicaciones.
- ...

Tolerancia a Fallos y Seguridad

- Tolerancia a fallos
  - Replicación de datos y de servicios
  - Mantenimiento de coherencia entre réplicas
- Tipología de los ataques a la seguridad:
  - Privacidad y confidencialidad.
  - Autenticación (*spoofing*).
  - Denegación de servicio.
- Modelos y herramientas de seguridad:
  - Cifrado: clave pública (RSA) y privada (DES).
  - Protocolos de seguridad: IPsec, SSL.
  - Certificados y firmas digitales: X.509.
  - Elementos de seguridad: Cortafuegos.
- Entornos de seguridad: p. ej. Kerberos.