Sistemas 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.
- Componentes de un sistema distribuido.

¿Qué es un sistema distribuido?

Distributed Systems. M. van Steen y A. S. Tanenbaum.

- *A distributed system is a collection of autonomous computing elements that appears to its users as a **single coherent system***

¿es siempre necesaria tanta transparencia?

Distributed Systems, Concepts and Design. G. Coulouris et al.

- *One in which **hardware or software** components located at networked computers communicate and coordinate their actions only by passing messages*

¿es un concepto hardware o software?

¡sin entrada en Wikipedia!

Fernando Pérez
María S. Pérez
José María Peña

S. distribuido: niveles físico y sistema

- Conjunto de nodos conectados por una red
- **Sin memoria compartida**
- Sin dispositivos compartidos
- Sin un reloj común
- Una copia del SO por nodo
- Fallos independientes
  - Error en componente hardware o en SO afecta solo a ese nodo
- Carácter heterogéneo
  - Nodos con distinto tipo de procesador y SO

S. distribuido: nivel de aplicaciones

- Aplicación (o servicio) distribuido
  - Procesos de la aplicación desplegados por distintos nodos
  - Necesitan comunicarse entre sí; También pueden requerir:
    - Compartir ficheros, acceso a dispositivos remotos, sincronización
    - Activar procesos remotos, gestión unificada de usuarios…
- SO local no proporciona esos servicios globales
- Capa software sobre SO local que los provea
- Objetivo de la asignatura: estudiar todo ese software
  - El "Sistema Operativo" del Sistema Distribuido
- Llevado al extremo, *Single System Image/View*:
  - El sistema se comporta exactamente igual que un sistema centralizado
  - Usuarios/aplicaciones no son conscientes del carácter distribuido
  - Muy complejo y no siempre deseable

Yet Another Definición de SD

Colección de procesos desplegados sobre un conjunto de nodos sin memoria compartida que interaccionan entre sí para realizar una determinada labor común

- Ejemplos de SSDD:
  - La web: un gigantesco SD que permite compartir información
  - Una solución de domótica
  - Una *Blockchain*
  - Un trabajo *MapReduce*

Ventajas de los Sistemas Distribuidos

Frente a sistema centralizado

- Economía: Buena relación rendimiento/coste.
- Capacidad de crecimiento: Escalabilidad.
- Alto rendimiento: Procesamiento paralelo.
- Soporte de aplicaciones inherentemente distribuidas.
- Fiabilidad y disponibilidad: Tolerancia a fallos.
- Carácter abierto y heterogéneo.
- Compartición de recursos y datos.

Desventajas de los Sistemas Distribuidos

Frente a sistema centralizado

- Necesidad nuevo tipo de software que incluya comunicaciones
- Red de interconexión introduce nuevos problemas:
  - Pérdida de mensajes y saturación.
  - Latencia puede provocar que al recibir un dato ya esté obsoleto.
- Más problemas de seguridad y confidencialidad
- Definición alternativa de SD:
  - "*Un sistema distribuido es aquel 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

1. *The network is reliable.*
2. *Latency is zero.*
3. *Bandwidth is infinite.*
4. *The network is secure.*
5. *Topology doesn't change.*
6. *There is one administrator.*
7. *Transport cost is zero.* Diversas interpretaciones:
   - Coste de empaquetar/desempaquetar datos al enviarlos/recibirlos
   - Coste económico de la infraestructura de red
8. *The network is homogeneous.*

Las tendremos en cuenta a lo largo del curso

- 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 (HPC):
  - Alternativa a costosos *supercomputadores* tradicionales.
  - "Sistema de computación distribuido"
- Servicios con alta disponibilidad y rendimiento.
- Sistemas distribuidos de gestión de bases de datos.
- Sistemas de procesamiento de datos masivos (*Big Data*).
- Aplicaciones multimedia.
- Sistemas industriales distribuidos y aplicaciones de control.
- Web: un enorme sistema distribuido.
- Ubicuos/IoT: automóviles, electrodomésticos, edificios...

Modelos de computación distribuida

- *Cluster Computing*
- *Grid Computing*
- *Volunteer Computing*
- *Utility Computing*
- *Cloud Computing*
- *Mobile Computing*
- *Ubiquitous Computing/Internet of Things*
- *Edge/Fog Computing*
- *Autonomic Computing*

Definiciones no excluyentes y algunas sin acuerdo general

Cluster Computing

- SD dedicado a ejecutar una aplicación buscando
  - Altas prestaciones y/o alta disponibilidad.
- Puede ejecutar varias aplicaciones mediante partición
- Más fuertemente acoplado que SD general
  - Incluso podría ofrecer *Single System Image/View*
- Poca dispersión geográfica
- Redes de alta velocidad
- Normalmente sistema con nodos homogéneos
- Carácter estático
- Uso habitual de componentes hardware estándar
- Necesidad de un planificador de trabajos

Grid Computing

- "*Cluster* virtual" sobre máquinas de varias organizaciones
  - Se crea para ejecutar aplicación y luego puede desaparecer
- 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
- Recursos no dedicados
  - *Grid* convive con SD de cada organización
- Sistemas heterogéneos
- Sistemas dinámicos

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

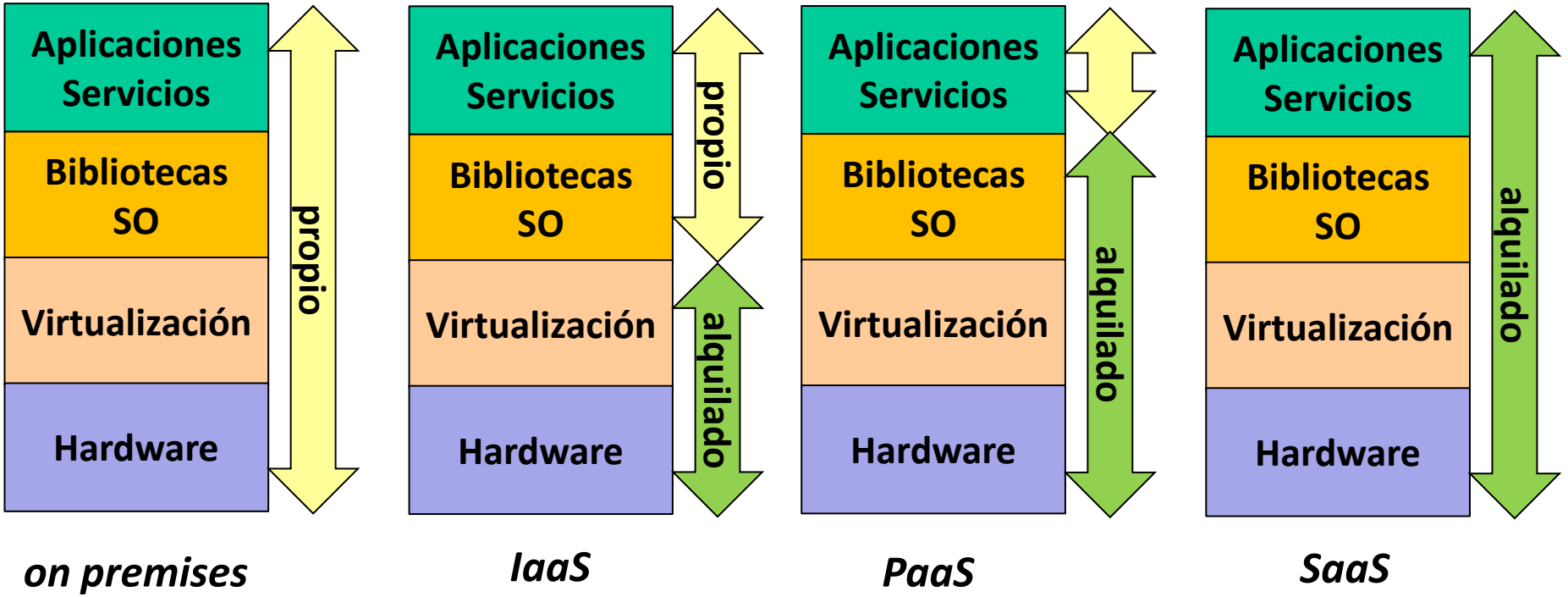
Utility Computing

- Computación como otro servicio público (agua, gas, luz…)
  - "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 usar un SD → computación *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."*
- Necesidad de esquema de tarificación

Cloud Computing

- Recursos HW y/o SW ofrecidos como servicio
  - Recursos virtualizados y dinámicamente escalables (elasticidad)
  - Pago por uso
- Modelos de servicio:
  - *Infrastructure as a Service* (AWS EC2, GCE, Azure VM)
    - Oferta dinámica de recursos HW virtuales según necesite el cliente
  - *Platform as a Service* (*Google App Engine, Azure App Service*)
    - Proporciona una plataforma de desarrollo de software
  - *Software as a Service* (*Google Workspace, Microsoft 365*)
    - Ofrece aplicaciones liberando del mantenimiento local de las mismas
- Modelos de despliegue:
  - *Público*: servicio público
  - *Privado*: servicio para solo una organización
    - Alojado y gestionado internamente (*on-premises*) o externamente
  - *Híbrido*: combinación de público y privado

Modelos de servicio



María S. Pérez
Fernando Pérez
José María Peña

Mobile Computing

- SD incluye dispositivos portátiles con acceso remoto
- Conectados de forma inalámbrica
  - a infraestructura del SD o formando redes *ad hoc*
- Usuario accede de forma remota a su organización
- Limitaciones en los recursos del dispositivo
- Control de consumo de energía del dispositivo
- Ancho de banda de comunicación variable
- Modo desconectado
  - Usuario puede seguir trabajando sin conexión
- Mayores amenazas a la seguridad y privacidad

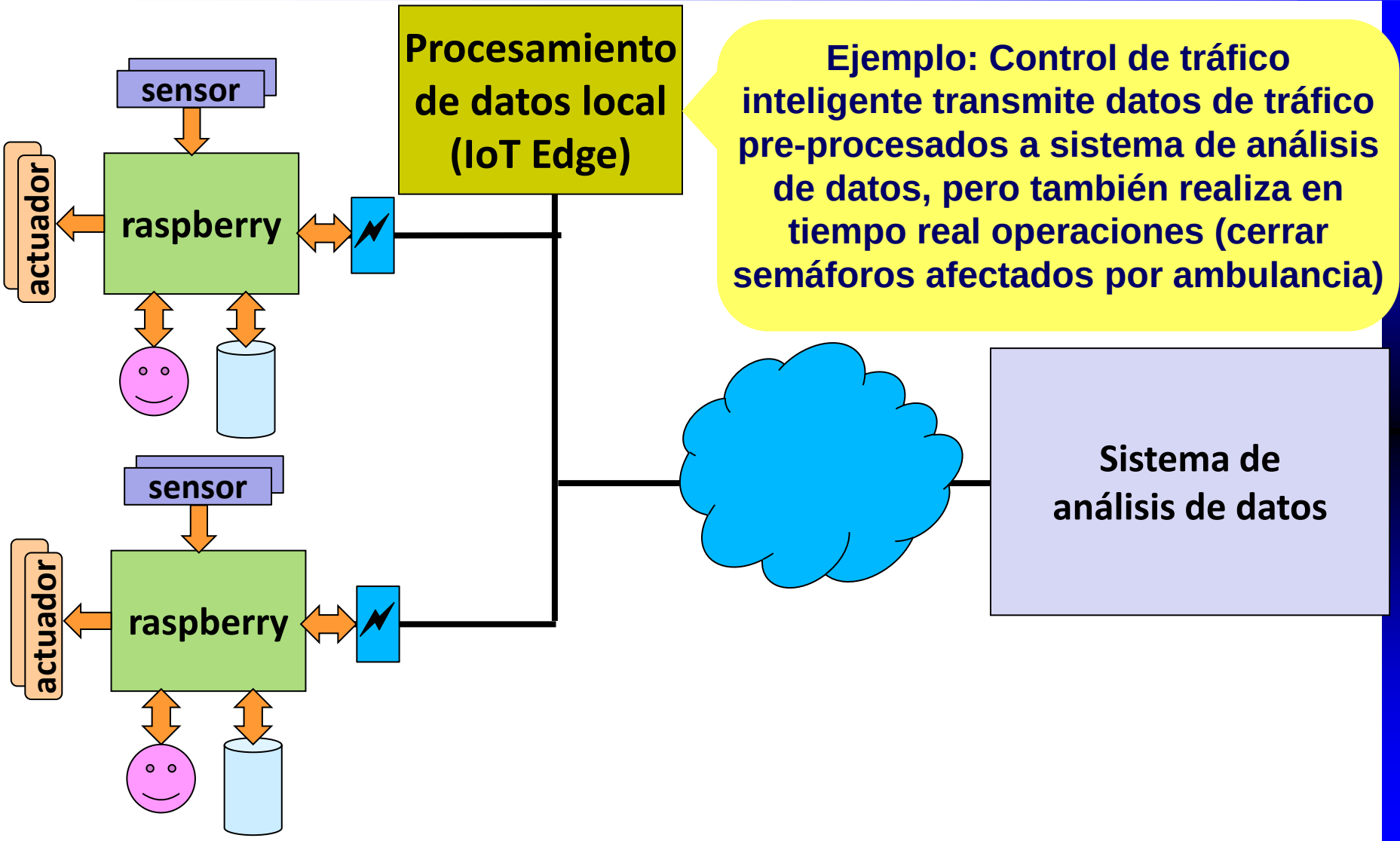
Computación ubicua/*IoT*

- Computadores omnipresentes, parte de ellos móviles
- Algunos empotrados en sistemas físicos, invisibles al usuario
- Otros portados por usuario (*Wearable Computing*)
- Conectados entre sí ofreciendo un valor añadido
- Plenamente integrados en el mundo para facilitar vida cotidiana
  - Pero permaneciendo prácticamente desapercibidos
- *Context-aware Computing*
  - Comportamiento de la aplicación depende del contexto
  - Ubicación, climatología, situación social del usuario…
- *Internet of Things* (IoT):
  - Objetos cotidianos (“cosas”) que se conectan a Internet
  - Puede verse como la implementación de la computación ubicua

Fog/Edge Computing

- Sistema *Cloud*: lejanía de fuentes de datos
  - Problemas de latencia, ancho de banda y tiempo de respuesta
- *Fog Computing*: "Entre la tierra y las nubes"
  - Despliegue de plataforma de computación cerca de fuentes de datos
  - Permite preprocesado de datos para reducir ancho de banda
    - Envía al sistema *cloud* los datos filtrados y preprocesados
    - Mejora escalabilidad
  - Y mejora tiempo de respuesta al disminuir la latencia
    - Sistema *fog* responde directamente a los eventos
- *Edge Computing.* En plataformas IoT:
  - Despliegue de nodos de computación cerca de los sensores
  - Preprocesamiento de los datos de los sensores antes de enviarlos
  - Respuesta en tiempo real

Edge Computing en IoT



María S. Pérez
Fernando Pérez
José María Peña

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 (*self-**):
  - Auto-configuración
  - Auto-reparación
  - Auto-optimización
  - Auto-protección
- Especialmente adecuado para computación ubicua:
  - Si cada componente requiriera atención aunque fuera mínima...

Objetivos de un Sistema Distribuido

- Transparencia
- Rendimiento
- Escalabilidad
- 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 Java RMI: deseable invocación métodos local ≠ remota
    - "*A Note on Distributed Computing*", Jim Waldo, 1994
    - Programador debe ser consciente de cuándo usa un método remoto:
    - Por la latencia que conlleva y la posibilidad de que falle

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

Escalabilidad

- Capacidad de mejora en prestaciones de sistema
  - Para hacer frente a un aumento de la carga que soporta
- Escalabilidad dinámica (elasticidad): auto-escalado
  - Prestaciones se van ajustando automáticamente a necesidades
- Diseño de SD debe evitar “cuellos de botella”:
  - Componentes centralizados
  - Estructura de datos centralizadas
  - Algoritmos centralizados

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...

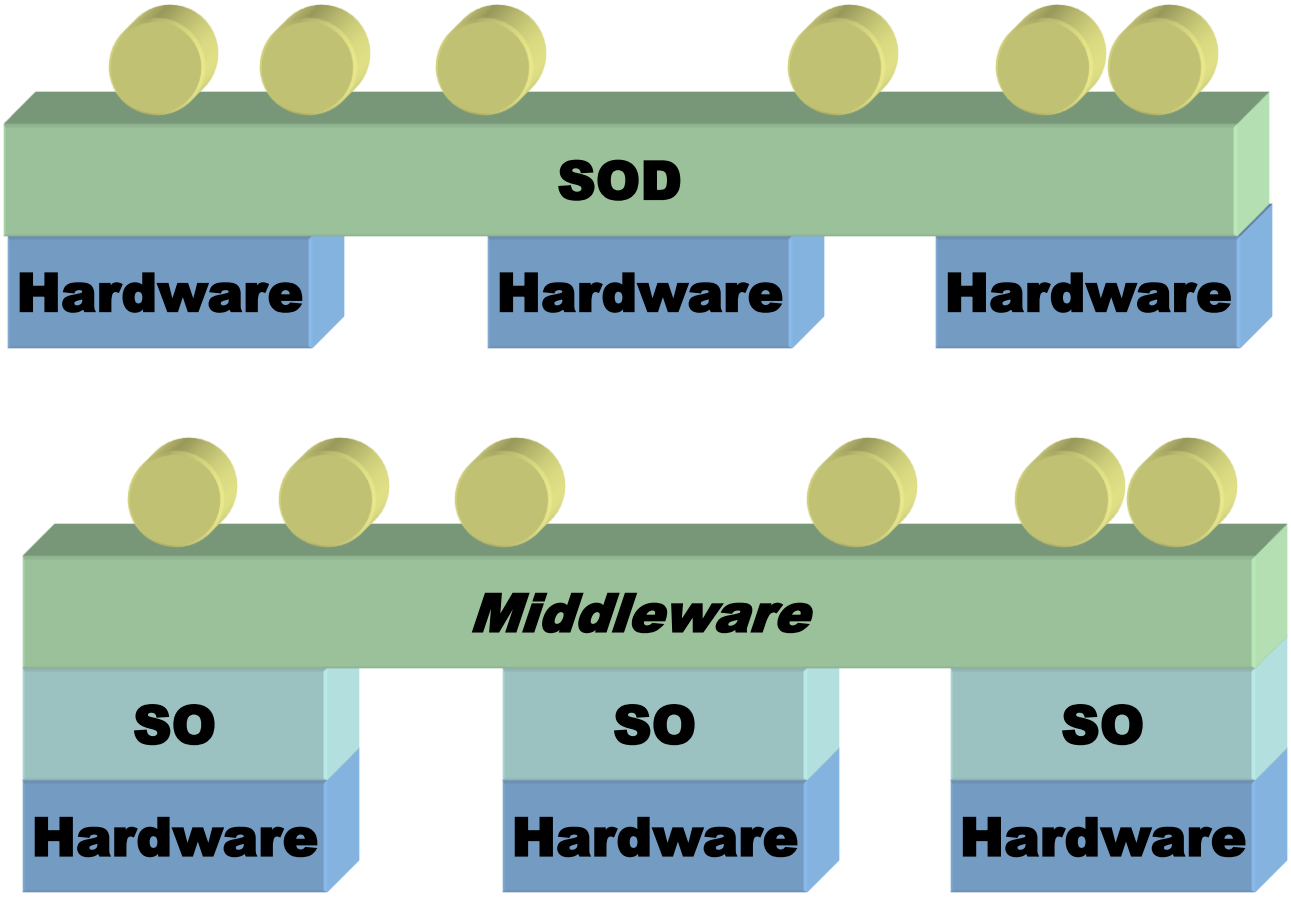
Fiabilidad

- Teóricamente: OR-lógico de sus componentes.
  - Funciona mientras haya un componente funcionando correctamente
- Sin embargo, a veces: AND-lógico de varios componentes.
  - No funciona en cuanto falle un componente
- Evitar componentes críticos (punto único de fallo).
- Uso de replicación
  - Mantenimiento de coherencia entre réplicas

Arquitectura software de los SSDD

- En una etapa se planteó la creación de SSOO Distribuidos
  - Crear desde cero un nuevo SO que gestione todo el SD:
    - ¿Cómo lograr exclusión mutua sin memoria compartida?
    - ¿Cómo tratar los interbloqueos sin un estado global?
    - ¿Cómo planificar procesos globalmente?
    - ¿Cómo gestionar un sistema de ficheros global?
    - Implicaciones de no reloj único, presencia de fallos o heterogeneidad.
  - Ejemplos final del siglo XX: Amoeba, Chorus y MOSIX
- No cuajó: ¿Tiramos a la basura los SSOO tradicionales?
- Mejor dejar que SO gestione recursos locales
- *Middleware*:
  - Capa de software que ejecuta sobre el sistema operativo local
  - Ofrece unos servicios distribuidos estandarizados.
  - No depende del hardware y sistema operativo subyacente.

SOD versus *Middleware*



Componentes de un Sistema Distribuido

Componentes $\approx$ Temario

- Arquitecturas y servicios de comunicación.
- Sistemas de ficheros distribuidos.
- Servicio de nombres.
- Servicios de sincronización y coordinación.
- Gestión de procesos.

Arquitecturas de comunicación

- Patrón de interacción entre componentes aplicación distribuida
- Cliente/servidor
- Editor/subscriptor
- Productor consumidor
- *Peer-to-peer*
- Arquitecturas para computación distribuida

Servicios de comunicación

- Paso de mensajes
  - Punto a punto (sockets)
  - Comunicación de grupo
  - Sistemas de colas de mensajes
- Llamada a procedimientos remotos (RPC)
- Invocación de métodos remotos (RMI)
- Servicios web
- Memoria compartida distribuida

Sistemas de Ficheros Distribuidos

- Estructura de un SFD
- Resolución de nombres
- Acceso a los datos
- Gestión de caché
- Gestión de cerrojos
- Casos de estudio: NFS y AFS
- Sistemas de ficheros paralelos:
  - Casos de estudio: Google File System y GPFS

Servicio de nombres

- Componente que gestiona nombre de recursos en SD
- Composición, distribución y replicación del espacio de nombres
- Servicio de páginas blancas vs. amarillas
- Casos de estudio: DNS y LDAP

Sincronización y Coordinación

- Sincronización de procesos compleja por
  - Falta de memoria compartida y fallos en algunos componentes
- Campo de desarrollo de algoritmos distribuidos:
  - Sincronización de relojes físicos y lógicos.
  - Exclusión mutua e interbloqueos.
  - Elección de líder.
  - Problemas de consenso.
  - Transacciones distribuidas.

Gestión de procesos

- Caracterización de la carga
- Estrategias de asignación de procesadores a procesos:
- Planificación de procesos
- Migración de procesos