Tema 1. Introducción

Sistemas Distribuidos

2022-2023

Definiciones clásicas

Tanembaum [2006]

Conjunto de computadores independientes que ante los usuarios del sistema parecen un solo computador

Colouris

Conjunto de computadores autónomos, unidos por una red, que ejecutan un software diseñado para dar una utilidad computacional integrada

Ejemplos

- Datos distribuidos (grandes volúmenes)
- Bittorrent (red *peer to peer* o P2P)
- *Grid computing* (SETI@home, Folding@home)
- *Clusters* (datacenters, Google, Amazon, Microsoft, etc.)
- *Cloud computing* (virtualización automatizada sobre los clusters)

Ventajas

- A veces no cabe elección (el problema es distribuído)
  - Compartir recursos físicos o lógicos
- Ventaja de SD sobre *supercomputador*
  - Precio
  - Escalabilidad
  - Rendimiento
  - Tolerancia a fallos (pero ¡cuidado! ¿se puede?)

Metas (y retos)

- Seguridad (privacidad, autenticidad, ataques...)
- Transparencia (acceso, localización, replicación, migración, fallos)
- No siempre la transparencia total es deseable (sistemas móviles oportunistas)
- Apertura (*openess*, uso de estándares, modularidad, componentes, interfaces abiertos)
- Escalabilidad (descentralización de servicios, datos y algoritmos)

Hacer SD es muy difícil

Falacias [Peter Deutsch]:

1. La red es fiable
2. La latencia es cero
3. El ancho de banda es infinito
4. La red es segura
5. La topología no cambia
6. Hay un administrador de la red
7. El coste del transporte es cero
8. La red es homogénea

¡NO!

Modelos (o paradigmas)

- Paso de mensajes
- Modelo cliente-servidor
- Llamadas a procedimientos remotos (o invocación de objetos remotos)
- Modelo *peer to peer*
- Agentes móviles
- Servicios *web*

Paso de mensajes

Paradigma fundamental. Subyace a todo.

- Primitivas básicas:
  - Enviar (`send`)
  - Recibir (`recv`)
- Modelos
  - Orientado a conexión (metáfora línea telefónica)
  - Sin conexión (metáfora sistema de correos)

Ejemplo: **sockets**

Modelo cliente servidor

Cada proceso que interviene en la comunicación tiene diferente rol

Servidor:	Cliente:
* Elemento pasivo. Espera peticiones.	* Elemento activo. Realiza peticiones.
* Realiza un servicio y generalmente proporciona una respuesta.	* Solicita servicio y espera respuesta

- Ejemplo: Muchos servicios en internet (Web, FTP, POP3, etc...)
- Implementaciones: sockets, RPC, RMI, SOAP, ...

Llamadas a procedimientos remotos (RPC)

IDEA

- Que un servicio se invoque mediante una llamada a una función (remota)
- Que la sintaxis sea la misma que una llamada local

```
float temperatura;
temperatura = obtenerTemperatura("Gijon");
printf("La temperatura en Gijon es de %fC\n",
       temperatura)
```

Invocación de métodos remotos (RMI)

- El concepto es el mismo que el de RPC
- Pero trasladado al paradigma de OOP
- Un objeto invoca un método que está en otro objeto remoto.
- Usando sintaxis de invocación local

```
double temperatura;
Sensor s = (Sensor) Naming.lookup("SensorServer");
temperatura = s.getTemperatura("Gijon");
System.out.println("La temperatura en Gijon es de " +
                   temperatura + "C");
```

Peer to peer (P2P)

En este modelo todos los procesos que se comunican son iguales entre sí.

- No hay distinción cliente/servidor.
- Aunque el rol sigue existiendo, cada proceso juega ambos roles.
- Facilita la *descentralización*.
- Muy utilizado en aplicaciones de descarga de archivos.

Agentes móviles

Un agente es un "paquete" que engloba código y datos.

- Se ejecuta en una máquina
- Pero puede migrar a otra (código y datos)
- En cada máquina por la que va pasando recoge más información
- Los agentes no intercambian mensajes

Es un modelo bastante experimental que plantea problemas de seguridad, y confianza

Servicios Web (REST)

- Es otra forma de invocación remota.
- El protocolo por el que se comunican cliente y servidor es el HTTP.
- El *verbo HTTP* se usa con fines *semánticos*.
- Los objetos se localizan mediante un *URL*.
- El formato de los mensajes es típicamente JSON.
- Es una forma de exponer servicios, antes accesibles vía web, para que sean más fácilmente consumibles por máquinas.
- Arquitecturas Orientadas a Servicios (o microservicios).