Software Distribuït - T1 - Introducció a les aplicacions distribuides.

Eloi Puertas i Prats

Universitat de Barcelona
Grau en Enginyeria Informàtica

13 de febrer de 2019

Exemples sistemes distribuïts

- Aplicacions web (Google, Banca electrònica, Botigues Electròniques...)
- Jocs Multiplayer (Massively multiplayer online games (MMOGS), Casinos...)
- Comunicacions (Skype, Whatsapp)
- Streaming i Compartició de fitxers Multimedia (Spotify, netflix, torrent...)

Tendències en sistemes distribuïts

1. Accés universal a Internet. Internet of Things (IOT)
2. Mobile & ubiquitous computing.
3. Sistemes Multimedia Distribuïts.
4. Software As A Service & hardware services! *Cloud Computing*

Necessitats dels sistemes distribuïts

1. Heterogeneïtat: Protocols, middleware, codi portable.
2. Accessibilitat: publicació de protocols, APIs, documentació.
3. Seguretat: Encriptació de dades sensibles.
4. Escalabilitat: Recursos HW i SW, controlar pèrdua de rendiment.
5. Tractament d'errors: Detectar, tolerar, recuperar-se d'errors.
6. Concurrència: Accés simultani a recursos compartits.
7. Transparència: Accés i localització del sistema transparent a l'usuari.
8. Qualitat del servei: Confiabilitat, seguretat i rendiment.

Arquitectura dels sistemes distribuïts

Sistema distribuït: ordinadors independents interconnectats que col·laboren per a realitzar una mateixa tasca.
Arquitectures: rols i responsabilitats

1. Model d'ordinador central amb terminals
2. Model client/servidor
3. Model Peer2Peer
4. Model microserveis

Model d'ordinador central amb terminals

- **Arquitectura**:
  - **Ordinador central**: dades + lògica
  - **Terminals**: presentació
- **Problemes**:
  - Saturació del servidor
  - Usuaris sense servei de transaccions

Model client/servidor

- **Arquitectura**:
  - **Servidor**: dades + lògica
  - **Clients**: presentació + lògica
  - Connexió sempre comença pel client, que ha de conèixer al servidor
  - Rols asimètrics.

Un servidor pot ser al seu torn, un client d'un altre servidor

- **Problemes**:
  - Problema d'escalabilitat.

Model Peer2Peer

- **Arquitectura**:
  - **Peer**: Node de computació/comunicació.
  - Rols simètrics, no hi ha diferència entre client i servidor
- **Problemes**:
  - Més complexitat: Lògica i dades totalment distribuïdes
  - Localització del servei (Bootstrap problem)

Model MicroServices

- **Arquitectura**:
  - Estructura una aplicació distribuïda com a una col·lecció de serveis.
- **Problemes**:
  - Més complexitat: Lògica i dades totalment distribuïdes
  - latència entre diferents serveis
  - Testing més difícil de provar totes les parts per separat i en conjunt.
  - Deployment més difícil (múltiples serveis s'han d'aixecar a l'hora)

Middleware

Middleware: Donar un nivell més alt d'abstracció per al desenvolupament d'aplicacions distribuïdes

- RPC
- Objectes Distribuïts RMI
- Components Distribuïts
- Serveis web

Crides a procediments remots

RPC (*Remote Procedure Calls*)

- **Programació no orientada a objectes**:
  - crides a procediments que s'executen en altres màquines
- **Programació Orientada a Objectes**: [finals '80]
  - invocació de mètodes a objectes que es troben en un altre servidor
  - **Objectes Distribuïts**: tots els objectes són alhora clients i servidors

Tecnologies d'objectes distribuïts

- **RMI** (*Remote Method Invocation*):
  - SO: independent
  - Llenguatge: dependent (Java)
- **DCOM** (*Distributed Component Object Model*):
  - SO: dependent
  - Llenguatge: independent (C, C++, VB...)
- **CORBA** (*Common Object Request Broker Architecture*):
  - SO: independent
  - Llenguatge: independent

Components distribuïts

Finals ’90: objectes es transformen en **components**

- **Servidor d’aplicacions**: contenidor de components
  - persistència automàtica
  - transaccions
  - cicle de vida
  - ...
- **Plataformes de components distribuïts**:
  - **J2EE** (*Java Platform Enterprise Edition*): JSP/Servlets
  - **Windows DNA** (*Windows Distributed interNet Applications Architecture*): ASP

Serveis web

Actualment:

- **Serveis web**:
  - Comunicacions: HTTP
  - Crides remotes: SOAP, XML-RPC, RESTful
  - Tec.Obj.Distribuïts: WSDL (*Web Services Description Language*)
  - Directori de serveis: UDDI (*Universal Description Discovery and Integration*)

Patrons Arquitectures multi-capa

Pas d'aplicació *monolítica* a aplicació **modular** estructurada en **capes lògiques** de:

- **usuari**: presentació, navegació, interacció.
- **negoci** / **lògica**: gestió d'esdeveniments sobre les dades, lògica del negoci, validació.
- **dades**: CRUD (*Create Retrieve Update Delete*), integritat ER.
- **sistema**: accés a fitxers, autentificació, errors, logs...

Capes físiques: Single-Tiered

Single-Tiered: 1 màquina amb 1 aplicació

- -: poc escalable, poc flexible, mono-usuari

Capes físiques: 2-Tier

2-Tier: client/servidor

- +: multi-usuari, un sol missatge per invocar una operació.
- -: capa negoci dividida entre el client i el servidor.

Capes físiques: 3-Tier

3-Tier: client / negoci / dades

- -: augment de missatges i complexitat
- +: cada màquina s'encarrega d'una capa

Capes físiques: N-Tier

N-Tier: n-client / n-negoci / n-servidor

- +: molt escalable

Tipus de client

- **Gruixut** (*Thick client*): 1/2 capes, bon maquinari ja que contenen molta funcionalitat.
- **Prim** (*Thin client*): capa lògica d'usuari, interfície senzilla d'introducció/extracció de dades (ex. navegador web).
- **Intermig** (*Pump Client*): capa lògica d'usuari + part capa de negoci (validacions simples).