

Concetti di applicazioni distribuite

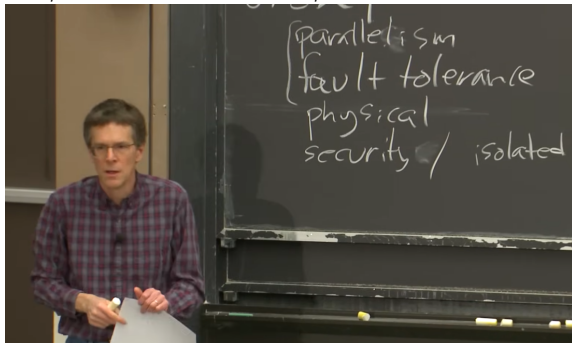
Gianluca Pironato

Introduzione

- ▶ Un'applicazione classica, in esecuzione in locale, mantiene interfaccia utente, gestione dati e logica applicativa su un unico dispositivo.
- ▶ Un'applicazione distribuita, al contrario, "sposta" gestione dati e/o logica applicativa su più dispositivi (nodi); deve comunque fornire una visione unica all'utilizzatore dell'applicazione, nascondendo l'eterogeneità dei componenti.
- ▶ Benefici della distribuzione:
 - ▶ scalabilità e tolleranza ai guasti (se un nodo è guasto, un altro può sostituirlo);
 - ▶ collaborazione, condivisione di risorse e riutilizzo dei moduli (si pensi a un server di logica applicativa a cui accedono due client con diversa UI).
- ▶ Svantaggi della distribuzione:
 - ▶ complessità di progettazione e sicurezza (minacce intrinseche della rete);
 - ▶ difficoltà nella gestione e nella previsione delle prestazioni.

Introduzione

Introduzione, fino al minuto 08:00, sui motivi della distribuzione:



Architettura client-server

- ▶ È un modello asimmetrico.
- ▶ Organizzazione:
 - ▶ Client: fa richiesta di un servizio.
 - ▶ Server: soddisfa la richiesta fornendo risorse (solitamente sono qui centralizzate logica applicativa e dati).
- ▶ Vantaggi:
 - ▶ Centralizzazione della gestione.
- ▶ Svantaggi:
 - ▶ Overhead di comunicazione (un solo server regge tutto il carico?).
 - ▶ Bassa scalabilità.

Client-server 2-tier

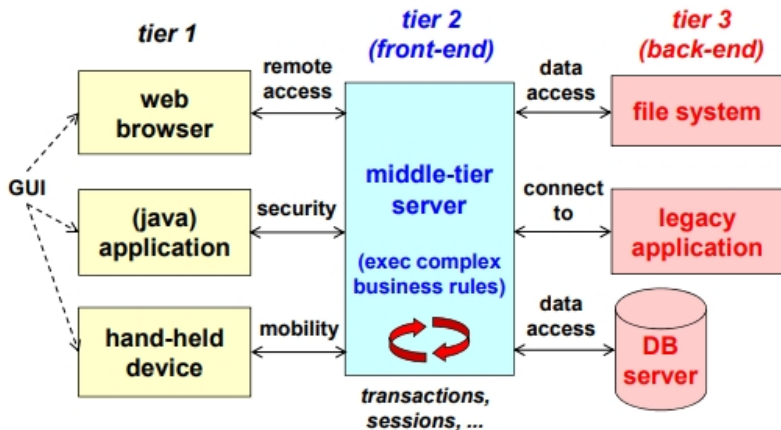
- ▶ I tre componenti (interfaccia utente, logica applicativa e dati) vanno distribuiti su due soli elementi. Come?
- ▶ **Client leggero e server pesante:** se il client si occupa solo della UI e il server mantiene logica applicativa e dati;
 - ▶ Sviluppo più semplice e carico concentrato su server.
- ▶ **Client pesante e server leggero:** se disponiamo solo i dati sul server e incarichiamo il client di UI e logica applicativa.
 - ▶ Distribuzione del carico, al costo di una gestione più complessa.

Client-server 3-tier

- ▶ Viene inserito un agente/broker tra client e server, così da seguire la suddivisione:
 - ▶ Livello 1: presentazione (interfaccia utente).
 - ▶ Livello 2: logica applicativa (elaborazione dati).
 - ▶ Livello 3: gestione dati (database).
- ▶ Vantaggi:
 - ▶ Migliore scalabilità e gestione.
 - ▶ Separazione delle responsabilità.
- ▶ Svantaggi:
 - ▶ Maggiore complessità di implementazione.
 - ▶ Overhead di comunicazione tra livelli.

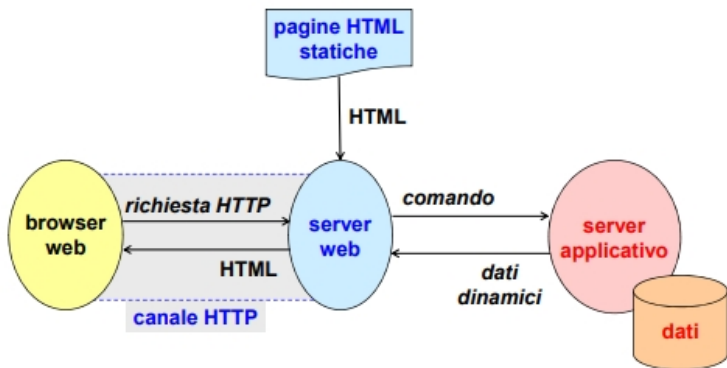
Client-server 3-tier

3-tier: esempio applicativo (custom)



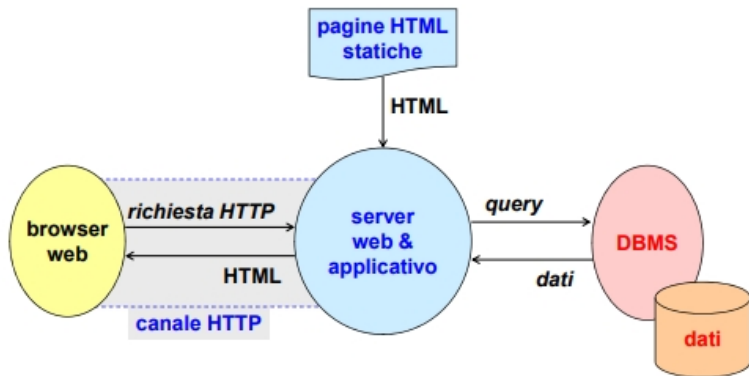
3-tier nel web

3-tier: modello web (caso I)



3-tier nel web

3-tier: modello web (caso II)



Architettura peer-to-peer

- ▶ È un modello idealmente simmetrico.
- ▶ Organizzazione:
 - ▶ I peer sono sia fornitori che consumatori di risorse.
 - ▶ Nessuna distinzione rigida tra client e server.
- ▶ Vantaggi:
 - ▶ Decentralizzazione e scalabilità.
 - ▶ Tolleranza ai guasti grazie alla ridondanza.
- ▶ Svantaggi:
 - ▶ Complessità nella gestione e nella sicurezza.
 - ▶ Difficoltà nel garantire consistenza e qualità del servizio (non c'è distribuzione sempre uguale e costante).

Conclusioni

- ▶ Le architetture distribuite offrono soluzioni per esigenze complesse di calcolo e condivisione.
- ▶ Ogni architettura presenta alcuni vantaggi:
 - ▶ Client-server: semplicità e centralizzazione.
 - ▶ 3-Tier: gestione efficiente e scalabilità.
 - ▶ Peer-to-peer: resilienza e decentralizzazione.
- ▶ Ricorda: la scelta dell'architettura dipende dai requisiti del sistema, non è sempre meglio distribuire.