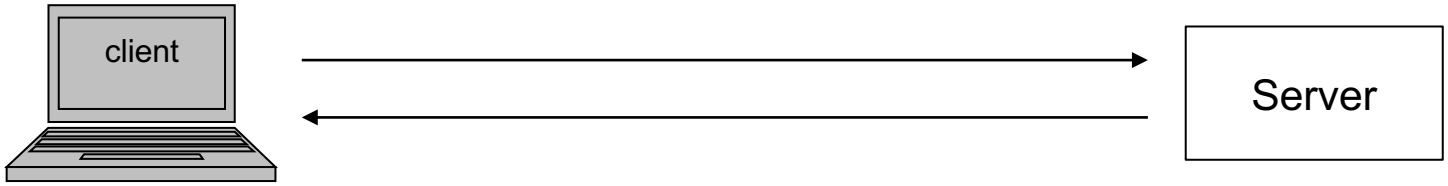
SUPSI

Architetture web

L. Sommaruga
P. Ceppi

Rev. 2022

client-server → architettura

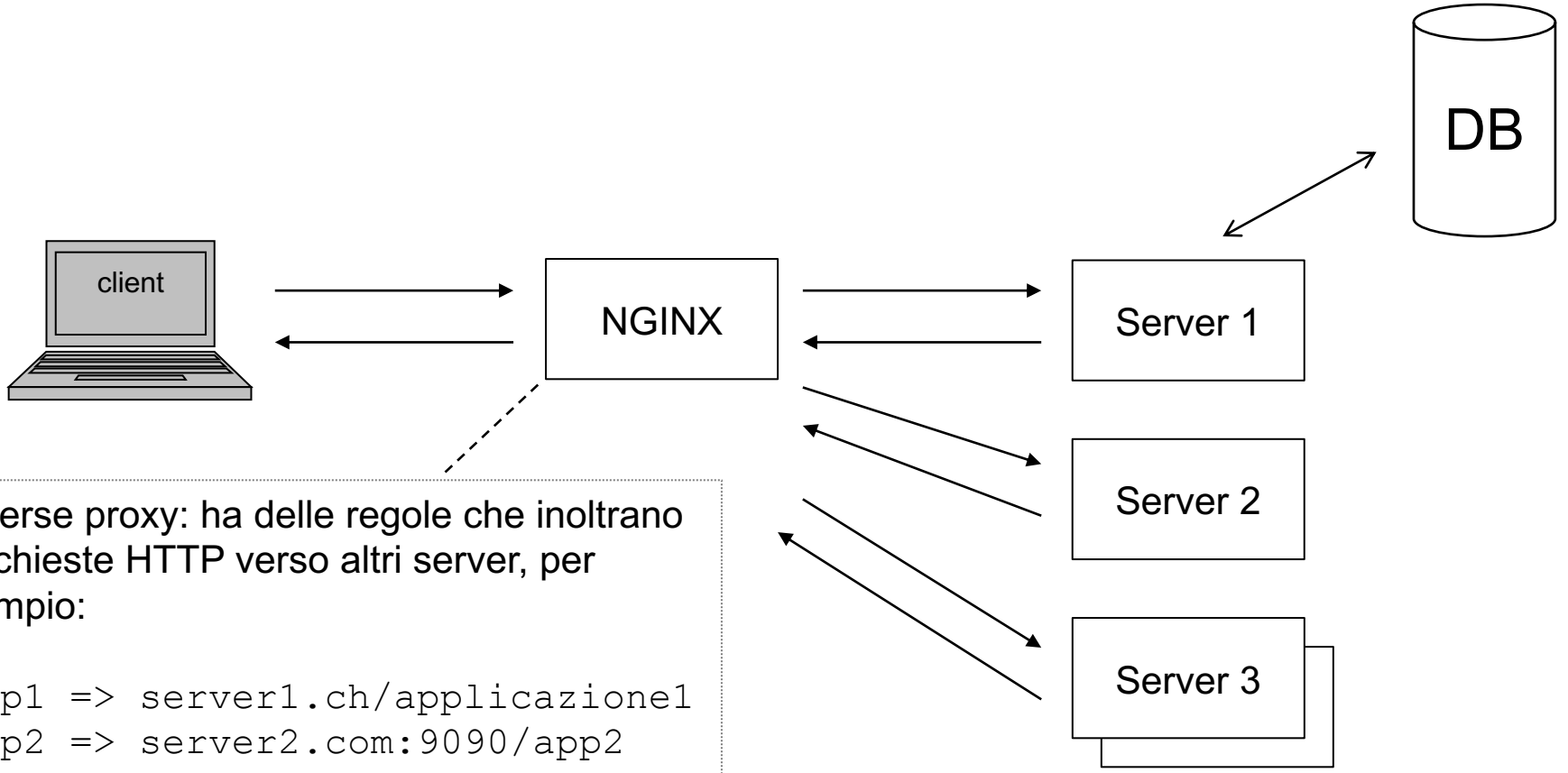


Cosa succede se la nostra applicazione ha un problema e viene killata?

Se vogliamo gestire più applicazioni/server sviluppati in maniera differenti l'uno dall'altro, ma avere un unico punto d'entrata, come facciamo?

Possiamo aggiornare/modifcare il server, cambiare il database, senza modificare la parte client o senza avere momenti di down?

client-server → architettura

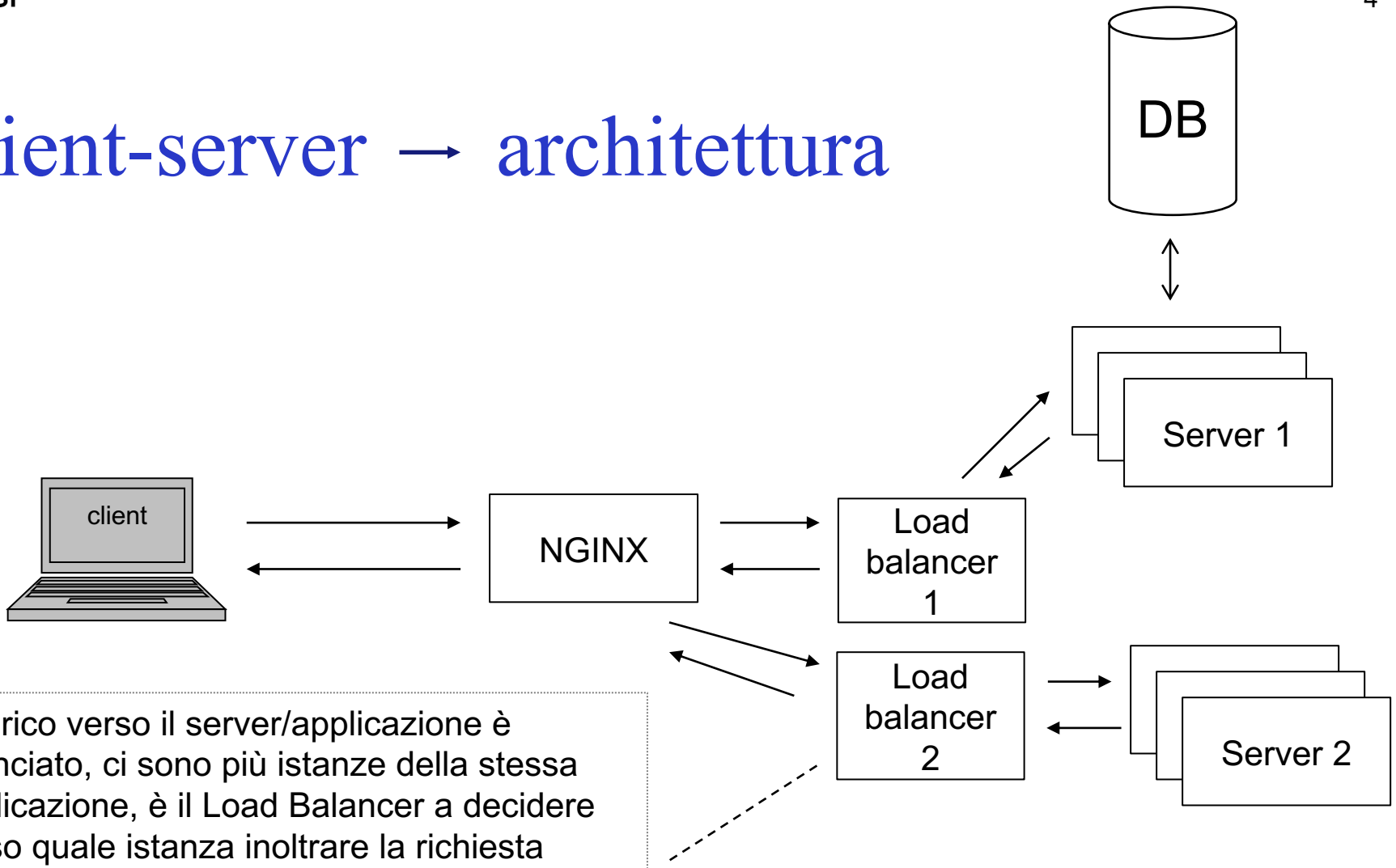


Reverse proxy: ha delle regole che inoltrano le richieste HTTP verso altri server, per esempio:

```
/app1 => server1.ch/applicazione1
/app2 => server2.com:9090/app2
```

unico punto di accesso verso l'esterno, ma acceda a risorse provenienti da più server

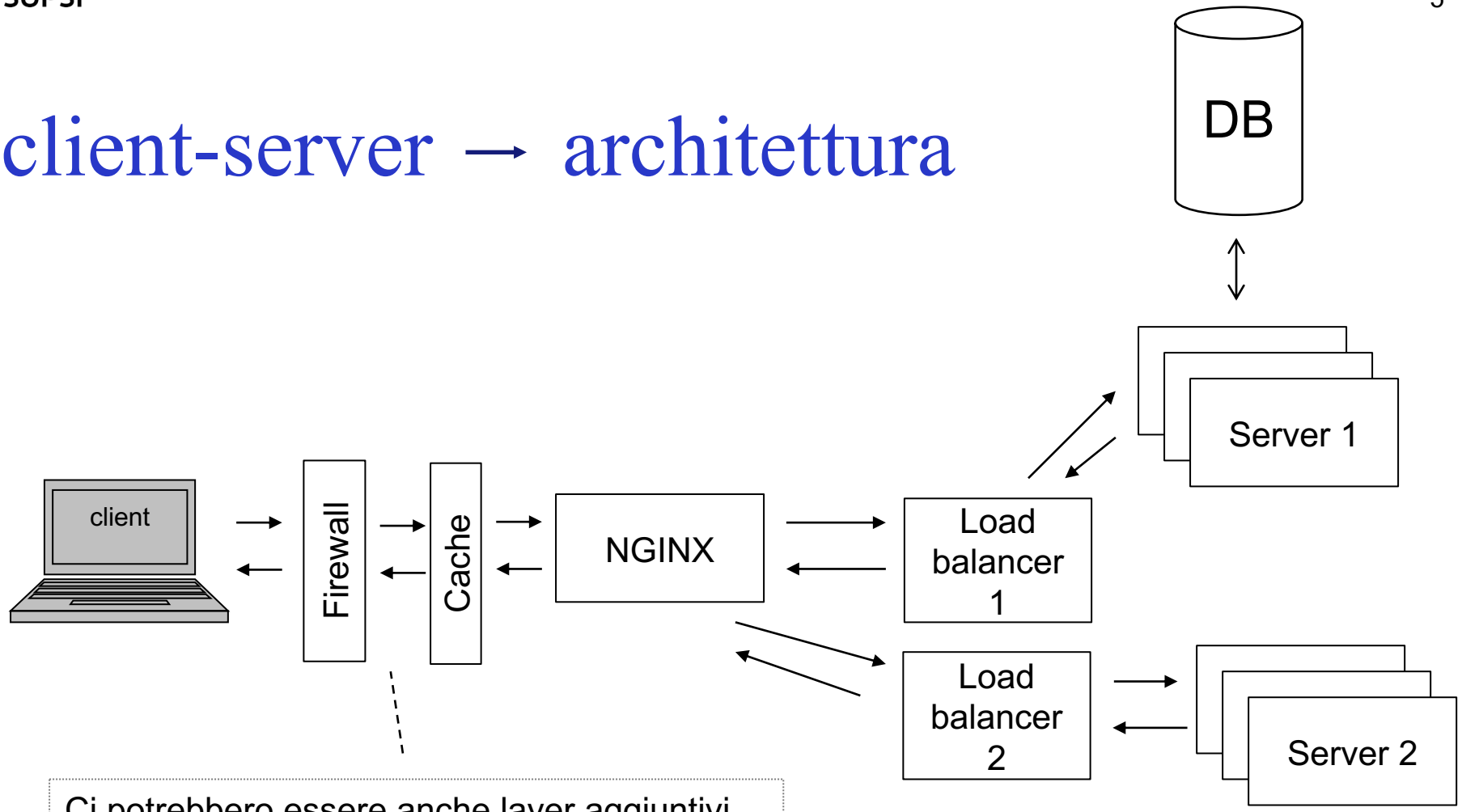
client-server → architettura



Il carico verso il server/applicazione è bilanciato, ci sono più istanze della stessa applicazione, è il Load Balancer a decidere verso quale istanza inoltrare la richiesta

Si riescono a gestire più richieste contemporaneamente e si è più tolleranti al fallimento di una delle istanze (l'app licazione è up 24h 7/7)

client-server → architettura



Ci potrebbero essere anche layer aggiuntivi nella *flusso* delle richieste HTTP: firewall, cache HTTP, …

Le richieste HTTP potrebbero subire delle modifiche, soprattuto negli header, mentre passano dai vari layer

Architetture web

- Architetture a N-livelli (tiers)
- Tre tipi separati di funzionalità:

  D - **Gestione dei Dati** (Data layer)

  L - **Logica di applicazione** (Business layer)

  P - **Presentazione**, interazione utente (Presentation layer)

- L'architettura del sistema determina se queste tre componenti risiedono su un singolo sistema (tier) oppure se sono distribuite su diversi tiers

Architettura N-tier

- Le varie funzionalità del software (D-L-P) sono logicamente e fisicamente separate ovvero suddivise su più strati o livelli software differenti in comunicazione tra loro

- Nonostante questa separazione fisica e logica, un'applicazione N-tier appare all'utente come un **unica unità**

- I diversi component dell'applicazione si interfacciano e interagiscono tra loro attraverso dei **middleware**:
  - Gestisce la comunicazione tra componenti
  - Garantisce disaccoppiamento tra i componenti
  - Permette sviluppo indipendente di un componente (S.O., linguaggio, piattaforma)
  - Semplifica la riconfigurazione del sistema

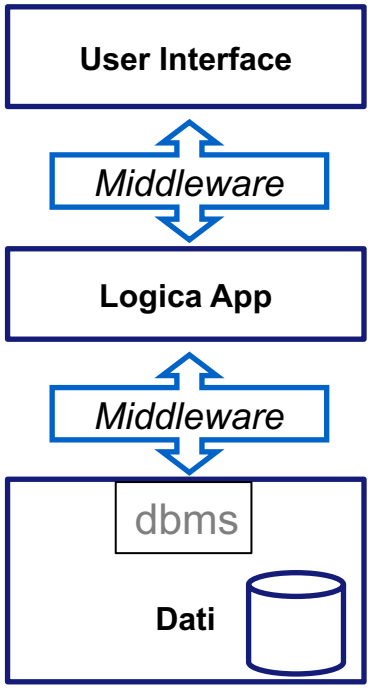
Presentation, Business e Data layer

- **Presentation layer**
  - È accessibile all'utente tramite il browser e consiste in un interfaccia utente che supporta l'interazione con l'utente
  - È sviluppata usando: HTML, CSS e Javascript
- **Business layer**
  - Accetta richieste dall'utente dal browser e le processa e determina a quali dati accedere e quale logica applicare per fornire delle risposte
- **Data layer**
  - Riceve tutte le richieste di accesso a dati e fornisce accesso ad essi

Schema architettura a 3 livelli

- Presentazione nel client
- Logica
- Dati ed accesso ai dati nel server

Partizionamento Logico



Partizionamento Fisico

Vantaggi architettura N-tier

- **Scalabilità**
  - permette al sistema e alle applicazioni di espandersi in modo scalabile senza modificare la struttura del sistema o gli algoritmi
- **Manutenzione**
  - permette di gestire e mantenere ogni componente senza influenzare gli altri componenti dell'architettura
- **Riusabilità**
  - alcuni componenti o parti dell'architettura possono essere riutilizzati per altre applicazioni
- **Sicurezza**
  - ogni tier può essere securizzato secondo i propri scopi
- **Tolleranza ai guasti**

Svantaggi

- **Complessità**
  - più layer e tier abbiamo, più abbiamo a che fare con un sistema complesso.
  - quando qualcosa non funziona può essere un problema
- **Tracciabilità**
  - tracciare in modo chiaro il flusso delle richieste e dei dati è più complicato. Ci vogliono dei sistemi e strumenti ad hoc.
- **Latenza**
  - aumento della latenza