

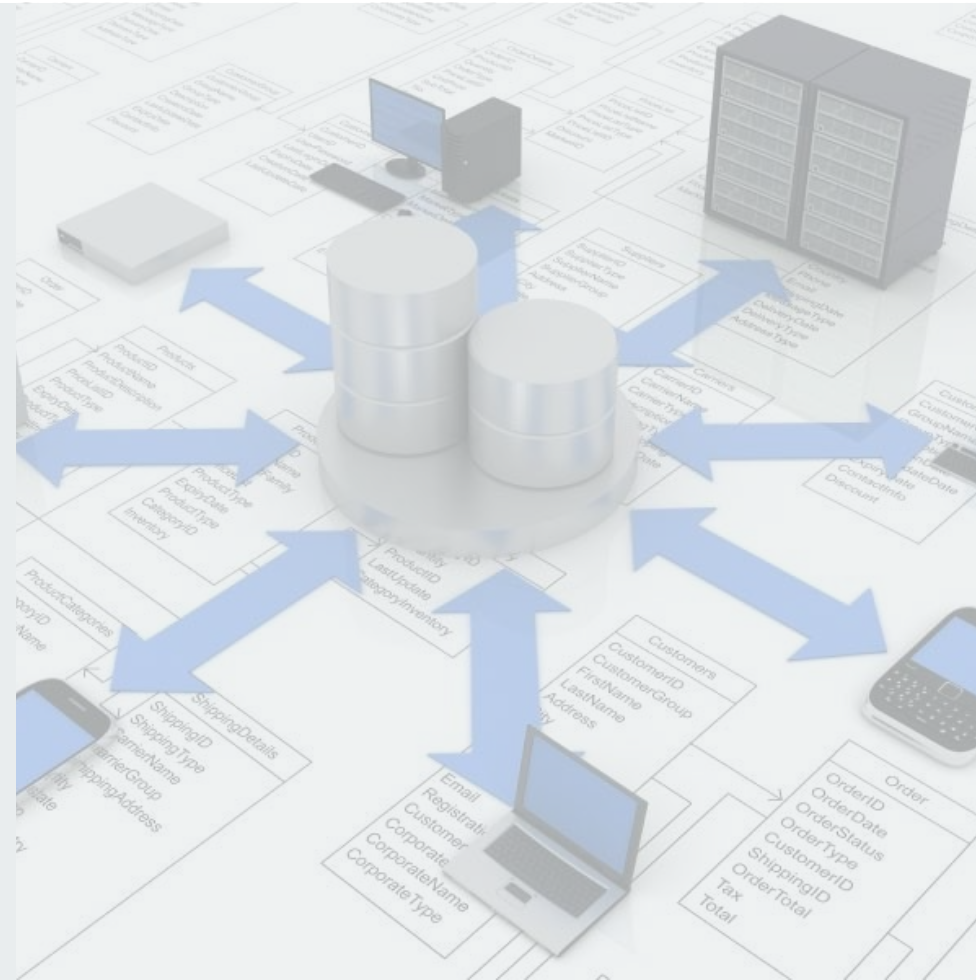
SISTEMI DISTRIBUITI

Evoluzione dei sistemi e modelli architetturali

Istituto Superiore
Antonio Scarpa
Motta di Livenza

Tecnologie e progettazione
di sistemi informatici e di telecomunicazioni | 2024/25

prof. Leonardo Sportiello





I sistemi distribuiti



La sincronizzazione dei dati tra i vari nodi è fondamentale. La distribuzione dei dati può portare a incongruenze, quindi è essenziale implementare meccanismi di consistenza come il modello **CAP** (*Consistenza, Disponibilità, Tolleranza alla Partizione*).

I sistemi distribuiti sono una rete di computer autonomi che collaborano per raggiungere un obiettivo comune, condividendo risorse e comunicando tra loro attraverso una rete. Ogni nodo (computer) può eseguire parti di un processo o elaborare dati in parallelo, rendendo il sistema più scalabile, affidabile e tollerante ai guasti rispetto a un sistema centralizzato.



Pianificare attentamente la gestione della complessità; è fondamentale comprendere che la progettazione, il debugging e la manutenzione saranno più complessi rispetto ai sistemi centralizzati.



I sistemi distribuiti **sono più vulnerabili** agli attacchi poiché **i dati** e le risorse **sono distribuiti su più nodi**. Implementare protocolli di sicurezza adeguati, crittografia e autenticazione è fondamentale per proteggere il sistema.



Protocolli di sicurezza Sistemi Distribuiti

- **SSL/TLS (Secure Sockets Layer/Transport Layer Security):** Utilizzato per garantire la sicurezza nella comunicazione tra i nodi attraverso la crittografia. SSL/TLS protegge i dati durante il loro trasferimento, prevenendo intercettazioni e manomissioni. Viene impiegato in servizi web, e-mail e molte altre applicazioni di rete.
- **Kerberos:** Un protocollo di autenticazione che utilizza la crittografia simmetrica per verificare l'identità dei nodi in una rete distribuita. Kerberos evita il rischio di attacchi di "replay" e assicura che solo i nodi autorizzati possano comunicare tra loro.
- **IPsec (Internet Protocol Security):** Utilizzato per proteggere il traffico di rete a livello IP, fornendo autenticazione e crittografia dei pacchetti di dati. IPsec è particolarmente utile per la creazione di reti private virtuali (VPN) sicure in sistemi distribuiti.
- **OAuth (Open Authorization):** Utilizzato per la gestione dell'accesso ai servizi distribuiti senza dover condividere le credenziali. OAuth consente agli utenti di concedere accessi limitati a risorse, particolarmente utile per l'accesso a servizi cloud e API.
- **PKI (Public Key Infrastructure):** Un sistema di gestione delle chiavi pubbliche e private che consente l'autenticazione sicura e la crittografia dei dati tra i nodi distribuiti. PKI è fondamentale per sistemi che richiedono firma digitale e cifratura asimmetrica.

Classificazione dei Sistemi Distribuiti




- **Sistemi di Calcolo con Cluster**
 - I sistemi di calcolo con cluster sono costituiti da un insieme di computer (nodi) collegati tra loro tramite una rete ad alta velocità. Ogni nodo in un cluster è una macchina fisica indipendente, ma i nodi lavorano insieme per eseguire compiti complessi come se fossero un unico sistema. I compiti vengono suddivisi tra i vari nodi per essere eseguiti simultaneamente; se un nodo in un cluster fallisce, altri nodi possono prendere il suo posto, garantendo la continuità del servizio; è possibile aggiungere più nodi per aumentare la capacità di calcolo.
- **Sistemi di Calcolo Grid**
 - I sistemi di calcolo grid ("griglie") sono costituiti da un insieme di risorse di calcolo distribuite geograficamente, che collaborano per risolvere problemi complessi. A differenza dei cluster, i grid possono includere risorse eterogenee, come supercomputer, server e persino dispositivi personali. I nodi di un grid sono spesso sparsi su un'area geografica vasta; condivisione di diverse tipologie di dati tra organizzazioni e utenti.
- **Sistemi Informativi Distribuiti**
 - I sistemi informativi distribuiti sono progettati per la gestione e la distribuzione di informazioni attraverso più nodi. Sono utilizzati principalmente per archiviare, gestire e processare dati su una rete di computer, fornendo accesso distribuito a tali dati. Trasparenza dei dati, tecniche di replicazione e sincronizzazione.
- **Sistemi Distribuiti Pervasivi**
 - I sistemi distribuiti pervasivi (noti anche come computazione ubiqua) si riferiscono a reti di dispositivi connessi che interagiscono in modo continuo con l'ambiente circostante per fornire servizi e informazioni agli utenti in tempo reale.


Presentazione di alcuni stili architetturali fondamentali per sistemi distribuiti – client/server e peer-to-peer

Lo sviluppo di sistemi software distribuiti è sostenuto dagli strumenti di

- **middleware**
 - una classe di tecnologie software sviluppate per aiutare gli sviluppatori nella gestione della complessità e della eterogeneità presenti nei sistemi distribuiti
 - uno strato software “in mezzo” – sopra al sistema operativo, ma sotto i programmi applicativi
 - ciascun middleware fornisce un’astrazione di programmazione distribuita
 - il middleware ha lo scopo di sostenere lo sviluppo dei connettori (sono anch’essi elementi software) per realizzare la comunicazione e le interazioni tra i diversi componenti software di un sistema distribuito




I connettori sono il luogo delle relazioni tra componenti, i cosiddetti *mediatori di interazioni* (ganci); ogni connettore ha una specifica di protocollo che definisce le proprietà, esistono diversi tipi: evento, procedura remota, broadcast.



il protocollo di un connettore comprende la specifica dei ruoli che devono essere soddisfatti – ad esempio **client e server**

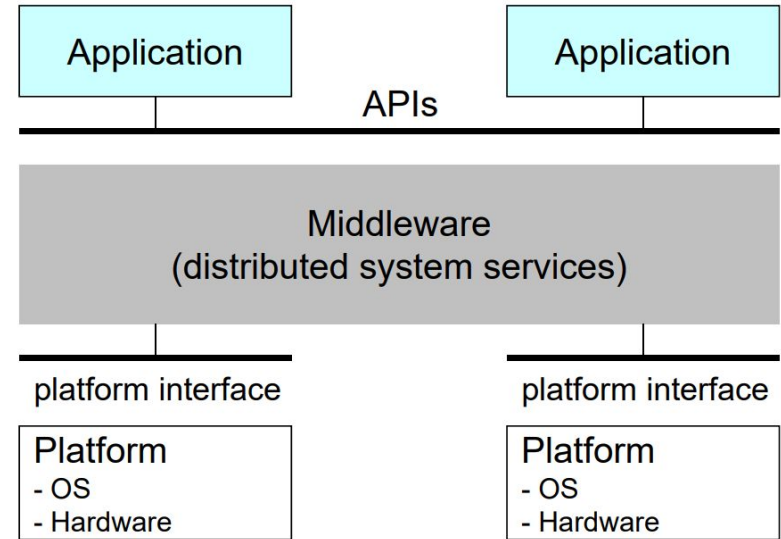
Middleware


Un servizio di **middleware** è un servizio *general-purpose* che si colloca tra piattaforme e applicazioni:  **rende facilmente programmabili i sistemi distribuiti**; ciascuno strumento di middleware offre una specifica modalità di interazione:

RPC: Remote Procedure Call, un meccanismo che consente ad un programma di chiamare funzioni o procedure su un altro nodo remoto come se fossero locali (XML-RPC).

Messaggistica asincrona: supporta la comunicazione tramite scambio di messaggi tra nodi. Questo avviene attraverso code di messaggi in cui un componente può inviare messaggi a un altro nodo, che li processerà in un secondo momento.

identifica hardware e software che risolvono problemi generali, in grado di eseguire programmi.



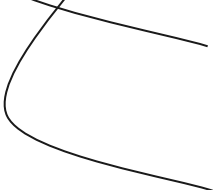
 sulla base di meccanismi di programmazione e API relativamente semplici; esempio **gRPC** e **XML-RPC**.

Varie famiglie di strumenti di middleware



1. middleware per componenti


2. middleware orientato ai servizi

- 
1. i componenti vivono in contenitori (application server) in grado di gestire la configurazione e la distribuzione dei componenti, e fornire ad essi funzionalità di supporto;
 2. flessibilità nell'organizzazione dei suoi elementi (servizi), in continua evoluzione;

L'applicazione di alcuni stili architetturali è sostenuta, dal punto di vista tecnologico, da opportuni strumenti di middleware; ad esempio, lo stile **C/S** può essere basato su **RPC**, lo stile **C/S a più livelli** sugli **application server (AS)**...

È chiaramente utile conoscere e comprendere queste relazioni per capire quale middleware utilizzare per realizzare un certo stile architetturale, e per capire come utilizzare al meglio un certo middleware.

Concetto di trasparenza nel Middleware

- Ciascuno strumento di **middleware** ha lo **scopo di mascherare qualsiasi tipo di eterogeneità**  comunemente presente in un sistema distribuito:
 - il middleware maschera sempre l'eterogeneità delle reti e dell'hardware;
 - alcuni strumenti di middleware mascherano eterogeneità nel sistema operativo e/o nei linguaggi di programmazione;
 - alcuni strumenti di middleware mascherano eterogeneità nelle diverse implementazioni di uno stesso standard di middleware.



in alternativa, il programmatore dovrebbe farsi esplicitamente carico di queste eterogeneità e di questi aspetti!

*Se utilizzato in modo opportuno, il **middleware** consente di affrontare e risolvere diverse problematiche* significative nello sviluppo dei sistemi distribuiti: consente di generare automaticamente tutti (o quasi) i connettori in questo modo, consente di concentrarsi sullo sviluppo della logica applicativa – e non sui dettagli della comunicazione tra componenti e della piattaforma hw/sw utilizzata.

Architettura Client-Server



Modello di coordinamento in un sistema distribuito.

Stabilisce:

- Chi può iniziare l'interazione
- Chi può rispondere
- Come trattare le condizioni di errore

Classificazione dei processi in:

- processi che richiedono servizi (client)
- processi che forniscono servizi (server)
- Il client richiede un servizio, il server esegue il servizio e rende disponibili i risultati al client

Architettura Client-Server



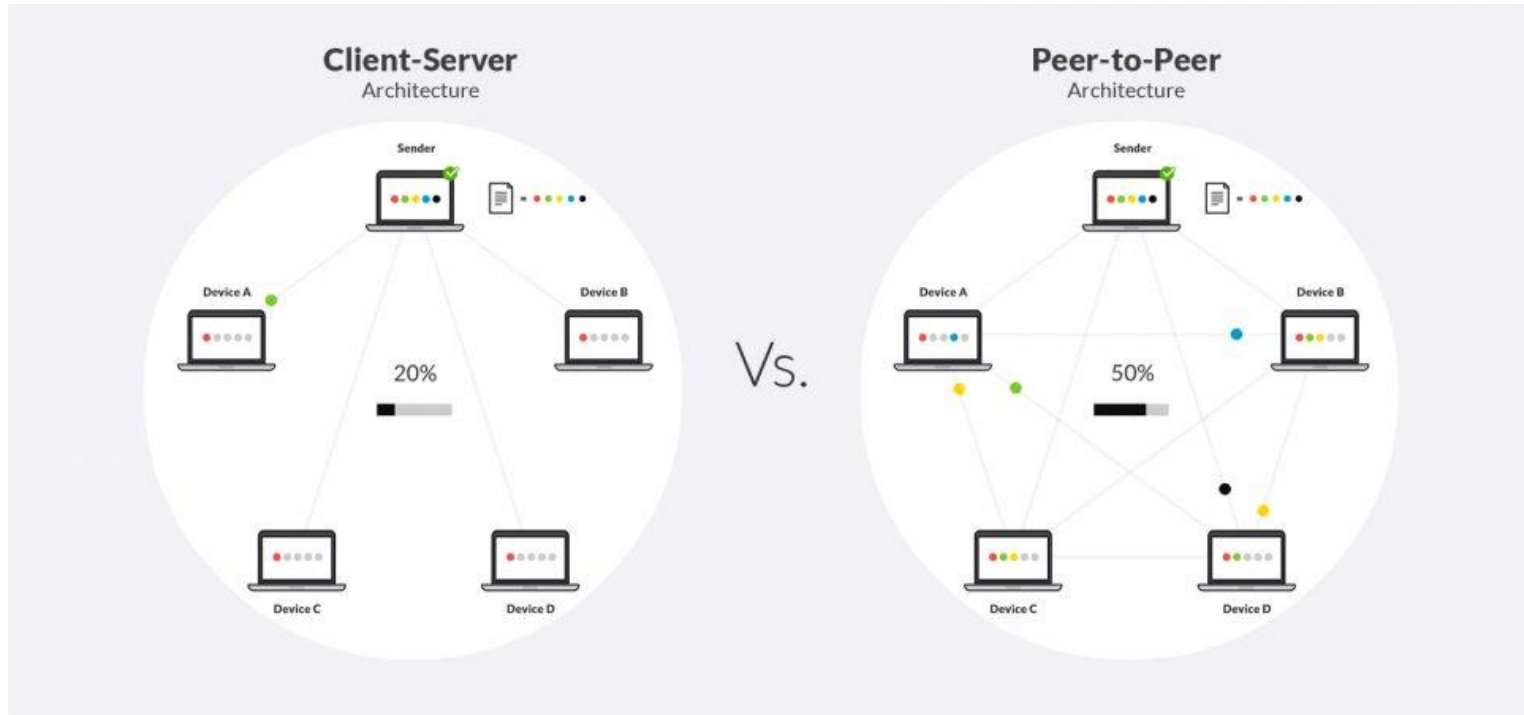
Funzioni del client

- fornisce un'interfaccia grafica (es. Windows) per ricevere richieste di servizio dall'utente.
- può eseguire, in parte, localmente il servizio
- prepara richieste per il/i processi servitori (utilizzo di SQL)
- utilizza strumenti di interprocess communication (IPC)
Es. Utilizzo di RPC (Remote Procedure Call)
- analizza i risultati inviati dal server

Funzioni del server

- Può solo rispondere alle richieste del client
- Può comunicare con altri server per rispondere alle richieste del client in modo invisibile al client
- Nasconde al client e all'utente la struttura del sistema

Architettura Client-Server - Peer to Peer | Differenza



Architettura Client-Server a Livelli (o "N-tier")



Livello di Presentazione (Presentation Tier):

- È il livello che interagisce direttamente con l'utente finale, fornendo un'interfaccia grafica o altri meccanismi di input/output.
- Tipicamente include applicazioni come browser web, app mobili o interfacce desktop.
- Comunica con il livello intermedio per richiedere dati e mostrarli all'utente.

Esempio: Il frontend di un sito web (HTML/CSS/JavaScript).

Livello Logico o Applicativo (Business Logic Tier):

- Contiene la logica di business, ossia tutte le regole e le operazioni che l'applicazione deve eseguire per soddisfare le richieste degli utenti.
- Questo livello può gestire la validazione, il calcolo e l'elaborazione dei dati.
- È il collegamento tra il livello di presentazione e il livello di dati.

Esempio: Un'applicazione server-side che elabora le richieste HTTP e interagisce con il database (Java, Python, PHP).

Livello Dati (Data Tier):

- Gestisce l'archiviazione e il recupero dei dati. Può essere composto da database relazionali o non relazionali, file system o altri tipi di archivi.
- Il livello dati è spesso protetto e accessibile solo dal livello logico.

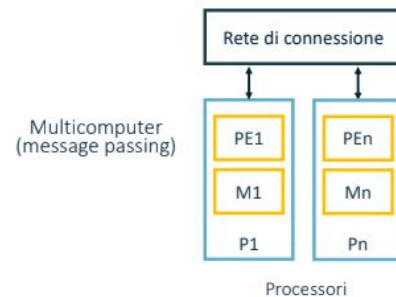
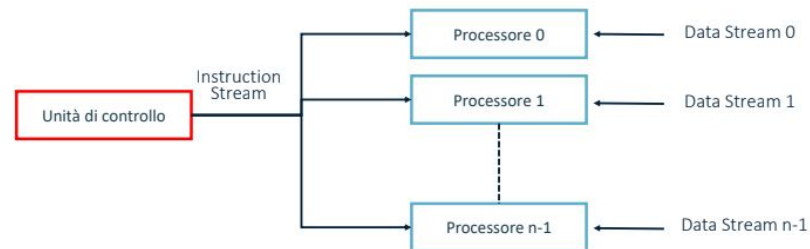
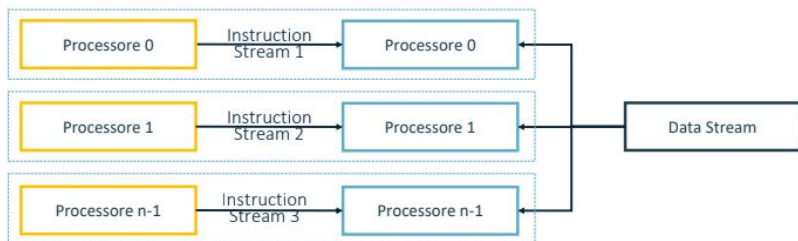
Esempio: Un database MySQL o un archivio NoSQL come MongoDB.

Tassonomia di Flynn

La tassonomia di Flynn è una classificazione dei sistemi di calcolo basata sul parallelismo dei flussi di istruzioni e di dati; consente di comprendere come i dati e le istruzioni vengono distribuiti e processati tra i nodi.

- **SISD (Single Instruction, Single Data)**
 - In un sistema SISD, un singolo processore esegue un'unica sequenza di istruzioni su un singolo flusso di dati.
 - Esempio: un calcolatore tradizionale che esegue un programma step-by-step senza parallelismo.
- **SIMD (Single Instruction, Multiple Data)**
 - Nei sistemi SIMD, una singola istruzione viene eseguita simultaneamente su più dati in parallelo. I nodi o processori eseguono lo stesso codice, ma ognuno opera su diversi dati.
 - Esempio: L'elaborazione grafica nelle GPU o i calcoli paralleli su matrici in applicazioni scientifiche.
- **MISD (Multiple Instruction, Single Data)**
 - In MISD, più istruzioni diverse vengono eseguite contemporaneamente sugli stessi dati.
 - Esempio: Sistemi critici, come gli algoritmi utilizzati nei controlli di volo.
- **MIMD (Multiple Instruction, Multiple Data)**
 - Nei sistemi MIMD, diversi processori eseguono istruzioni indipendenti su dati diversi in parallelo; supporta la massima flessibilità e scalabilità.
 - Esempio: Un cluster di server in un sistema cloud computing, dove ogni server gestisce richieste o processi differenti e su dati separati.

Tassonomia di Flynn



Tassonomia di Flynn

