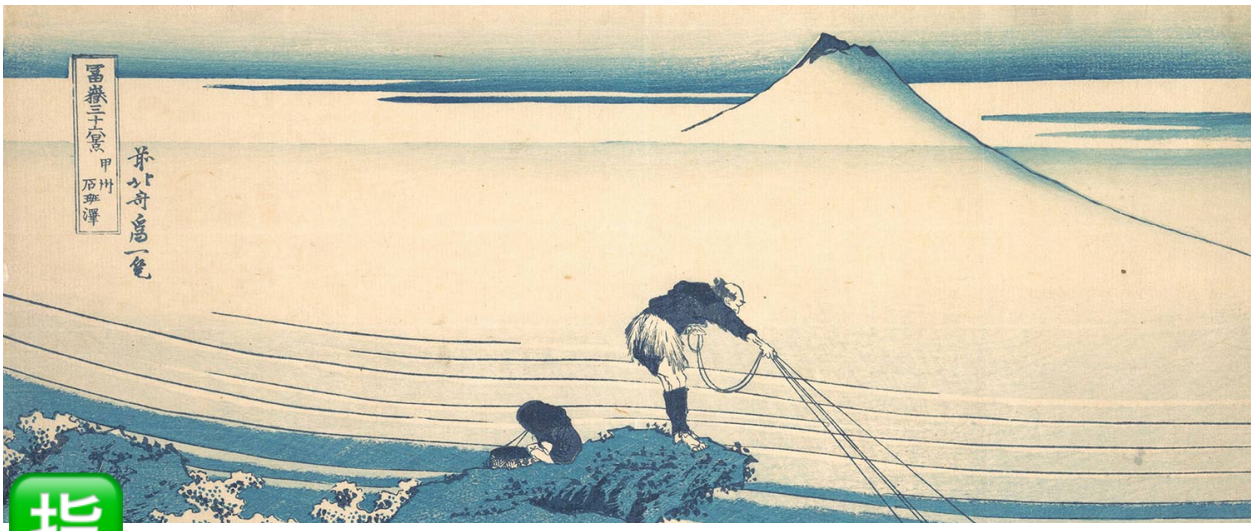


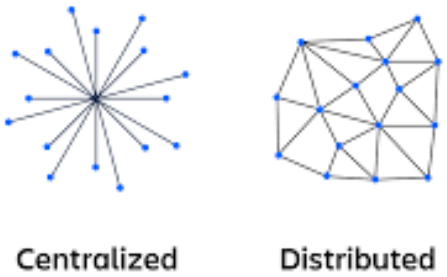
Sistemi Distribuiti

Tags	SIS DIS

Sistema centralizzato → tutto locale e tutto sviluppato intorno ad una sola unità di calcolo e il tutto è nello stesso posto locale un esempio è un solo server situato nella rete locale al quale poi tutti vanno a richiedere dati

Sistema distribuito → I sistemi distribuiti sono una tipologia di sistema informatico in cui componenti software e hardware,
dislocati su diversi nodi di una rete (come computer, server, dispositivi mobili, ecc.), collaborano e comunicano tra loro per raggiungere un obiettivo comune. L'idea alla base di un sistema distribuito è di permettere l'elaborazione e la condivisione di risorse in modo efficiente e affidabile, superando le limitazioni di un singolo sistema isolato.

Meglio un sistema è sviluppato e scalabile, più vuol dire che è distribuito



Dove possiamo trovare sistemi distribuiti →

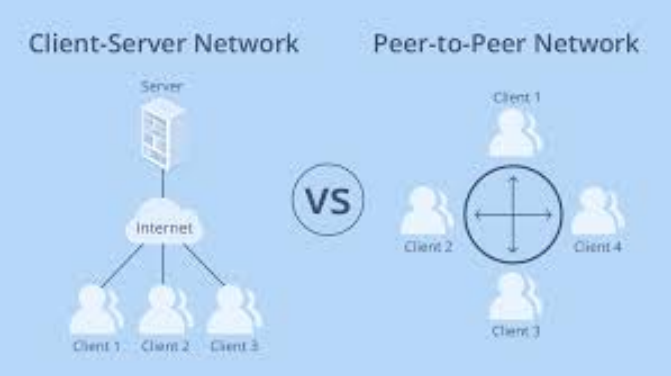
I sistemi distribuiti sono utilizzati in molte applicazioni pratiche per migliorare l'efficienza, l'affidabilità, la
scalabilità e la disponibilità dei servizi informatici. Alcuni esempi di utilizzo includono:
• Server Web e Cloud Computing: Distribuzione delle risorse su più server per bilanciare il carico e garantire alta disponibilità.
• Reti di Sensori: Utilizzate per monitorare ambienti e raccogliere dati in tempo reale.
• Sistemi di File Distribuiti: Come HDFS (Hadoop Distributed File System), che consentono
l'archiviazione e l'accesso a grandi quantità di dati in modo distribuito.
• Applicazioni di E-commerce: Sistemi che gestiscono transazioni e dati di utenti distribuiti su più
server per migliorare la sicurezza e la velocità.
• Blockchain e Sistemi Finanziari: Utilizzano nodi distribuiti per garantire l'integrità e la trasparenza delle transazioni.

All’interno di un sistema centralizzato se è presente un guasto sono cazzi amari al contrario in quello distribuito gli altri nodi si adatteranno e andranno a svolgere anche le funzioni che svolgeva il nodo guasto

Tipi di sistemi distribuiti in base all’architettura

Sistema Peer to Peer (P2P) → Sistema all'interno del quale tutti i nodi fungono sia da client che da server quindi tutti si occupano di tutto ciò fa in modo che all'interno della rete non sia necessario un server ma nel momento del bisogno i singoli end-point fungeranno da server e in altri momenti da client

Client-Server → Architettura tradizionale dove i client richiedono servizi a uno o più server
centralizzati.



Tipi di sistemi distribuiti in base alla coerenza dei dati

Sistemi distribuiti fortemente consistenti → Sistemi all'interno dei quali tutti i nodi sanno tutto di tutti

Sistemi distribuiti eventualmente consistenti → I dati che circolano possono essere talvolta incoerenti ma alla fine tutto quadra

Tipi di sistemi distribuiti in base alla topologia della rete:

Bus → Tutti i nodi condividono un unico canale di comunicazione.
Anello (Ring) → I nodi sono collegati in una configurazione ad anello, dove ogni nodo è
connesso esattamente a due altri nodi.
Stella (Star) → Un nodo centrale agisce come hub per la comunicazione con altri nodi.
Maglia (Mesh) → Ogni nodo è connesso direttamente a uno o più altri nodi, migliorando la
ridondanza e l'affidabilità.

Categorie di sistemi distribuiti

Sistemi di calcolo a cluster → Ogni nodo lavora come indipendente ma ciò permette a tutti di lavorare contemporaneamente e ad alte velocità

All'interno di questi sistemi possiamo trovare prestazioni elevate destinate a calcoli complessi come Big Data e AI

Sistemi Greed → I nodi del sistema non necessariamente si trovano nella stessa posizione fisica ma anche in altre aree geografiche ma lavorano lo stesso in simultanea per esempio laboratori scientifici.

A differenza dei cluster, i grid possono
includere risorse eterogenee, come supercomputer, server e persino dispositivi personali.

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.

Caratteristiche Principali:

• Accesso e Condivisione di Dati Distribuiti: Consentono a utenti e applicazioni di accedere e
condividere dati distribuiti su diversi nodi in modo trasparente.
• Coerenza dei Dati: Devono gestire la coerenza dei dati tra diversi nodi, utilizzando tecniche come la
replica e la sincronizzazione.
• Affidabilità e Disponibilità: Progettati per garantire l'accesso continuo ai dati anche in caso di
guasti o disconnessioni di nodi.

Esempi:
• Sistemi di Database Distribuiti: Come Apache Cassandra, che memorizza e gestisce grandi volumi
di dati distribuiti su molti nodi.
• Sistemi di File Distribuiti: Come Hadoop Distributed File System (HDFS), utilizzato per archiviare e
gestire grandi quantità di dati su cluster distribuiti.

Sistemi Pervasivo → Rete di nodi che interagiscono tra loro in maniera dinamica e in tempo reale un esempio è lo smartwatch che nel momento in cui ci alleniamo prima va a salvare le info nel server poi le elabora e ce le rifornisce.

Sistemi Distribuiti → Rete di nodi autonomi che collaborano per raggiungere un obbiettivo comune

- scalabile
- affidabile
- tollerante ai guasti
- sincronizzato → + importante

Benefici dei Sistemi Distribuiti

1. Scalabilità: I sistemi distribuiti permettono di aggiungere nuove risorse (nodi) facilmente per
   aumentare la capacità di elaborazione o di archiviazione.
2. Affidabilità e Tolleranza ai Guasti: In un sistema distribuito, la perdita o il malfunzionamento di un
   singolo nodo non comporta necessariamente il fallimento dell'intero sistema. Ridondanza e replica
   dei dati possono essere utilizzate per garantire la continuità del servizio.
3. Disponibilità: I sistemi distribuiti possono essere progettati per garantire alta disponibilità,
   riducendo il tempo di inattività grazie alla ridondanza e alla distribuzione delle risorse.
4. Prestazioni: I sistemi distribuiti possono migliorare le prestazioni suddividendo le attività di
   elaborazione su più nodi, riducendo così il carico su ciascun nodo individuale.
5. Flessibilità e Modularità: È possibile sviluppare e aggiornare componenti del sistema
   indipendentemente, senza interrompere l'intero servizio.

Svantaggi dei Sistemi Distribuiti

1. Complessità: La progettazione, implementazione e gestione di sistemi distribuiti è molto più
   complessa rispetto ai sistemi centralizzati. Devono essere affrontate problematiche come
   sincronizzazione, coerenza dei dati, tolleranza ai guasti e sicurezza.
2. Problemi di Comunicazione: Poiché i nodi di un sistema distribuito comunicano attraverso una
   rete, sono soggetti a ritardi di trasmissione, perdita di pacchetti, congestione della rete e altri
   problemi legati alla comunicazione.

3. Sicurezza: Aumenta la superficie di attacco, poiché ogni nodo rappresenta un potenziale punto di
   ingresso per malintenzionati. La sicurezza dei dati e delle comunicazioni diventa cruciale.
4. Gestione della Consistenza dei Dati: Garantire che tutti i nodi abbiano una vista coerente dei dati
   può essere complesso, soprattutto in presenza di guasti o ritardi di rete.
5. Costi di Implementazione: Richiede risorse hardware e software significative, oltre a
   un'infrastruttura di rete affidabile e sicura.

Protocolli di sicurezza Sistemi Distribuiti

Proprio per le criticità di sicurezza che caratterizzano i sistemi distribuiti vengono adottate misure di sicurezza per fare in modo di tutelare la sicurezza della rete e del sistema.

SSL/TLS → HTTPS → Certificato(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) → Responsabile per esempio nel momento in cui decidiamo di accedere ad un dato account di una piattaforma tramite account google

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.

Middleware

L’obbiettivo di un sistema distribuito è quello di offrire una visione unica di tutto il sistema come se effettivamente non fosse diviso per fare ciò è stata studiata una organizzazione basata su più strati:

- Livello Superiore → utenti e applicazioni
- Livello Intermedio → strato software
- Livello Basso → S.O.

Il livello intermedio, interfaccia tra piattaforma e applicazione, è detto middleware.

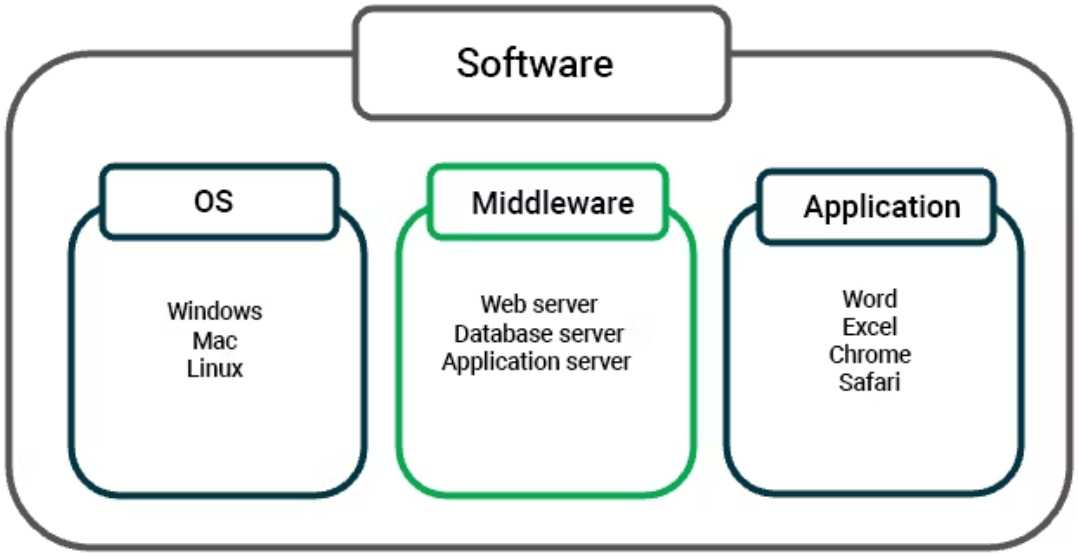
Il middleware è come un "traduttore" o un "messaggero" che aiuta due programmi diversi a parlarsi e a lavorare insieme. Immagina di avere un videogioco e un controller: il middleware si assicura che il controller possa inviare i comandi giusti al videogioco, anche se sono costruiti in modo diverso. In poche parole, è un aiutante che fa funzionare meglio le cose tra due programmi

Classe di risorse software sviluppata per aiutare gli sviluppatori nella gestione della complessità e della eterogeneità presente nei sistemi distribuiti

- Si collega e si pone tra software e piattaforme
- Collegano :
    - hardware e software
    - software e software
    - hardware e hardware

Il middleware è di tipo software gestito attraverso API relativamente semplici come :

gRPC / XML-RPC



Architettura client server a livelli

- 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

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.