0.TPS

1) Sistemi distribuiti

Le architetture dei sistemi informativi si sono sviluppate ed evolute negli anni, passando da schemi centralizzati a modelli distribuiti.

Sistema centralizzato: i dati e le applicazioni risiedono in un unico nodo elaborativo.

Sistema parallelo: è un insieme di elementi di elaborazione che comunicano e cooperano per risolvere velocemente problemi (devono cooperare per forza, non come i sistemi distribuiti che possono scegliere).

Sistema distribuito: si parla di sistema distribuito quando le applicazioni risiedono su più host che collaborano tra di loro (**elaborazione distribuita**) oppure quando il patrimonio informativo è ospitato su più host (**base di dati distribuita**).

Ci sono diverse definizioni di sistema distribuito ma nessuna è pienamente soddisfacente. Possiamo definire (in modo astratto) che un sistema distribuito è costituito da un insieme di applicazioni (logicamente indipendenti) che collaborano per raggiungere determinati obbiettivi comuni attraverso un infrastruttura hardware e software.

Ruoli delle applicazioni:

-**Client**: quando l'app utilizza i servizi messi a disposizione da altre applicazioni;

-**Server**: quando l'app fornisce i servizi usati da altre applicazioni;

-**Actor**: quando l'app assume sia ruolo di Client che di Server.

Altre definizioni di sistema distribuito

-Un sistema distribuito consiste di un insieme di calcolatori indipendenti che appaiono all'utente del sistema come un singolo calcolatore" (Tanenbaum, 1995);

-un sistema in cui l'elaborazione delle informazioni è distribuita su più calcolatori anziché centralizzata su una singola macchina;

-un sistema di elaborazione in cui un numero di componenti coopera comunicando in rete;

-un sistema in cui i componenti hardware o software posizionati in calcolatori collegati in rete comunicano e coordinano le proprie azioni solo tramite lo scambio di messaggi [Coulouris & Dollimore];

-un sistema in cui il fallimento di un calcolatore di cui nemmeno conosci l'esistenza può rendere inutilizzabile il tuo calcolatore [Lamport].

-Vantaggi e Svantaggi dei sistemi distribuiti

Vantaggi:

Affidabilità	Il sistema distribuito è in grado di sopravvivere in caso di guasto grazie alla ridondanza intrinseca (in caso di piccoli guasti).
Integrazione	È la capacità del sistema di integrare componenti hardware e software eterogenei tra loro. Il linguaggio utilizzato per favorire lo scambio di informazioni web tra sistemi operativi diversi è XML.
Trasparenza	Vedere il sistema come un unico sistema di elaborazione, e non come un insieme di componenti. Le forme di trasparenza sono di accesso, di locazione, di concorrenza, di replicazione, ai guasti, alla migrazione, alle prestazioni e di scalabilità.
Economicità	Miglior rapporto prezzo/qualità rispetto ai sistemi centralizzati.
Apertura	Aperto in termini di hardware e software, in modo da avere ampliabilità, portabilità e interoperabilità.
Connettività e collaborazione	Vantaggi economici grazie alla condivisione di risorse hardware e software.
Prestazioni e scalabilità	Con l'aggiunta di nuove risorse e servizi, tutti i componenti subiscono un miglioramento delle prestazioni e un aumento delle richieste che possono soddisfare
Tolleranza ai guasti	Il sistema funziona anche in presenza di piccoli guasti.

Svantaggi:

Produzione di software	I programmatori hanno dovuto cambiare il loro modo di programmare e aggiornarsi con lo studio di nuovi linguaggi.
Complessità	I sistemi distribuiti sono più complessi di quelli centralizzati a causa della loro struttura hardware.
Sicurezza	Bisogna aumentare il livello di sicurezza rispetto ai sistemi centralizzati per evitare l'accesso a chi non ne ha diritto.
Comunicazione	Il trasferimento a distanza delle informazioni richiede nuove tipologie di sistemi di telecomunicazione e si ha un aumento sempre maggiore di richieste di bande trasmissive.

---

2)Modelli architetturali hardware

-SISD: un'unica unità di elaborazione esegue un unico flusso di dati. *Es macchina di Von Neumann*

-**SIMD:** più flussi di dati vengono eseguiti in contemporanea da un singolo flusso di istruzioni.

Es. adatta per realizzare calcoli vettoriali e matriciali

-**MISD:** è un'architettura parallela in cui si eseguono più istruzioni in un singolo flusso di dati.

Non ancora implementata

-**MIMD:** è un'architettura parallela in cui si eseguono più istruzioni su più flussi di dati (LAN).

Si dividono in MIMD a memoria fisica condivisa (multiprocessor) e in MIMD a memoria privata (multicomputer).

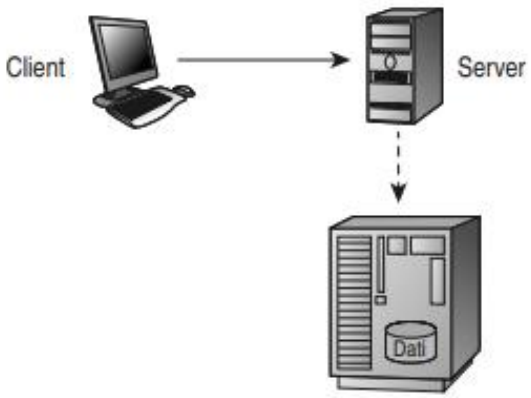
-CLUSTER: sono un insieme di computer interconnessi che appaiono all'utente come una singola risorsa computazionale e le varie componenti sono dedicate al funzionamento d'insieme. La potenza di un cluster è la somma dei singoli computer che la costituiscono. È diversa rispetto a una rete di PC

per la centralizzazione fisica delle macchine, per la velocità di trasferimento dati e per la presenza di un app management.

Architetture software

-**Architettura a terminali remoti**: le operazioni vengono effettuate da un'unica entità centrale alla quale sono collegati dei terminali remoti che visualizzano, inviano e ricevono le informazioni;

-**Architettura client-server**: i client hanno una loro capacità di elaborazione e richiedono un servizio al server, che esaudisce la richiesta, la elabora e invia la risposta al client. I client possono collaborare tra di loro e quest'architettura è la più adatta per far comunicare entità non omogenee;



-**Architettura cooperativa**: è l'evoluzione dell'architettura client-server. Si basa su entità autonome che esportano e richiedono servizi (secondo il modello di sviluppo a componenti) per la programmazione lato server (programmazione ad oggetti);

-**Architettura WEB-centric**: il web server viene reso come il centro del sistema distribuito, dove i client accedono per ottenere dei servizi;

-**Architettura completamente distribuita**: si oppone all'architettura web-centric e viene utilizzata nei sistemi dove è necessaria la cooperazione di gruppi di entità uguali (groupware) che dialogano tra di loro offrendo i propri servizi;

-**Architettura a livelli**: per alleggerire il carico elaborativo dei server sono state introdotte le applicazioni multilivello e sono stati introdotti gli strumenti middleware, uno strato software che si trova tra il S.O. (sopra) e i programmi applicativi (sotto). Esso ha lo scopo di realizzare la comunicazione tra i diversi componenti software di un sistema distribuito.

Per middleware si intende un insieme di programmi informatici che fungono da intermediari tra diverse applicazioni e componenti software.

	Architetture MIMD		OS Network	Middelware
	Multiprocessori	Multicomputer		
Gradi di trasparenza	Molto alto	Alto	Basso	Alto
Unico SO su tutti i nodi	SI	SI	NO	NO
NR. di copie del SO	1	N	N	N
Forma di comunicazione	Memoria condivisa	Messaggi/ memoria condivisa	Files	Modelli specifici
Gestione delle risorse	Globale centralizzata	Globale distribuita	Per nodo	Per nodo
Scalabilità	NO	Modesta	SI	Variabile
Apertura ai diversi SO	CHIUSO	CHIUSO	APERTO	APERTO

3)Modelli di comunicazione

Object based

In questo modello ogni risorsa viene vista come un'oggetto e ognuno di questi viene identificato univocamente. Quando un processo necessita di una risorsa invia un messaggio contenente la richiesta all'oggetto corrispondente: questo la esamina, la elabora eseguendo l'operazione richiesta e invia un messaggio di risposta (come nel modello client server, ciascun oggetto può essere il gestore di una risorsa o contemporaneamente, l'utente di un'altra risorsa).

Client server

Il modello client server è il modello architetturale più importante. È formato da un insieme di host che forniscono i servizi, i server, e dai client, che usufruiscono dei servizi. Non sono gli host a essere client o server ma sono i processi e essi possono diventare client o server in base alle esigenze (dato che su un host possono essere in esecuzione più processi, un host può essere contemporaneamente sia client che server).

Questo modello permette di gestire applicazioni molto complesse grazie alla sua evoluzione. Alcuni tipici servizi utilizzati dalle architetture client-server sono **TELNET, HTTP, FTP,** SMTP, NFS…

Funzionamento:

Il client per comunicare con un server deve usare il protocollo TCP/IP. Inizialmente deve connettersi al socket dell'host dove il server è in esecuzione, specificando l'IP della macchina e il numero di porta in cui il server è in ascolto (socket).

Il server, quindi, rimane in ascolto su una determinata porta finché il client non crea una comunicazione con il socket (specificando la porta) ed esegue le richieste del client, rispedendo i risultati.

Comunicazione:

-unicast: il server comunica solo con un client alla volta e accetta una richiesta di connessione solo se nessun altro client è già connesso;

-multicast: al server possono essere connessi più client contemporaneamente. Se la richiesta va a buon fine, sposta la richiesta su una nuova porta, lasciando la precedente libera in attesa di altre connessioni (manda un thread che soddisfa la richiesta).

Numeri di porte

0-1023=numeri noti, servono per determinati servizi;

1024-49151=numeri registrati, solo sotto richiesta;

49152-65535=dinamici, sono liberi.

http: 80 o 8080.

La trasmissione dei pacchetti tra server e client avviene con due modalità (approcci):

-Connection less: più veloce ma meno affidabile (**protocollo UDP**);

-Connection oriented: meno veloce ma più affidabile (**protocollo TCP**). Anche se il pacchetto prende strade diverse, si ha sempre la certezza dell'arrivo di esso grazie all'Ack. Questa verifica però rallenta il processo.

Protocollo UDP: è fatto per i flussi e funziona per mezzi che fanno pochi errori (streaming). Si usa quando si utilizza un solo pacchetto.

Protocollo TCP: crea un canale di comunicazione con ACK. Invia pacchetti uno alla volta e deve attendere la risposta del destinatario. Se il pacchetto non arriva a destinazione, lo rinvia.

Livelli e strati

Le architetture client-server sono normalmente organizzate a livelli: ciascun livello funziona da server per i suoi client nel livello precedente e da client per il livello successivo ed è organizzato in base al tipo di servizio che fornisce.

I compiti del client-server si possono suddividere in 3 livelli:

-**Resource management layer**: si occupa della gestione delle risorse (è composto dall'insieme delle procedure);

-**Application logic layer**: si occupa dell'elaborazione delle informazioni;

-**Presentation layer**: rende le informazioni leggibili per l'utente.

Storia delle architetture a livello

Architettura a un livello (1 tier): a partire dagli anni 70, le architetture si riducevano a un solo mainframe, collegato a degli "stupidi" terminali: l'elaborazione veniva effettuata dall'elaboratore centrale, mentre i terminali si occupavano delle fasi I/O.

Architettura a due livelli (2 tier): a partire dagli anni 80, sono nate le architetture client server. I 2 livelli, quindi, erano il livello client e quello server. Riconosciamo 2 sottocategorie:

-modello thin-client, dove il server è responsabile della logica applicativa e della gestione dei dati (Application logic layer e resource management layer), mentre il client è responsabile dell'esecuzione del software di presentazione (presentation layer);

-modello thick-client, dove il server è responsabile della gestione dei dati (resource management layer) mentre il client è responsabile della presentazione e della logica applicativa (application logic layer e presentation layer, viene spostata parte dell'applicazione sul client).

Si è passati dal thin al thick-client per aumentare la potenza di calcolo del pc ma c'è un però, infatti questo tipo di architettura è poco scalabile dato che i server deve gestire la connessione e lo stato della sessione di ciascun client (questo porta alla limitazione del numero di client che possono essere gestiti contemporaneamente).

Architettura a tre livelli (3 tier): nato negli anni 90, a ogni livello corrisponde uno strato architetturale (presentation tier, middle tier e data tier). L'introduzione del middleware porta con sé numerosi vantaggi, in termini di prestazioni (+distribuzione della quantità di elaborazione, -velocità). Il sistema è facilmente scalabile ed è molto sicuro. Altri svantaggi, oltre alla velocità, sono la loro progettazione, lo sviluppo e l'amministrazione.

Architettura a n tier: è una generalizzazione del modello 3 tier, dove vengono scomposti e introdotti un numero qualunque di livelli e server intermedi (suddivide i compiti dei vari strati). Questa scomposizione viene chiamata multi-tier.

4) Applicazioni di rete

Le applicazioni di rete (o distribuite) sono quelle applicazioni che vengono utilizzate dall'utente, in cui il compito del programmatore è quello di utilizzare le primitive di comunicazione messe a disposizione dai protocolli degli altri strati. Queste applicazioni di rete vengono implementate dal livello delle applicazioni del modello ISO/OSI e TCP e sono costituite da un insieme di programmi che vengono eseguiti su due o più computer contemporaneamente (implementano le applicazioni di rete).

Modello ISO/OSI (pila, stack)

Viene usato come modello di riferimento per consentire la comunicazione aperta tra diversi sistemi ed è formato da una pila di protocolli.

7	APPLICAZIONI	*Input e output*
6	PRESENTAZIONE	*Garantisce che i dati vengano trasmessi nei formati standard*
5	SESSIONI	*Organizza la connessione tra i sistemi che fanno parte della comunicazione*
4	TRASPORTO	*Unione tra livelli applicazione e trasporto, vengono usati TCP o UDP*
3	RETE	*Viene assegnato un indirizzo IP univoco*
2	COLLEGAMENTO	*Indirizzamento dell'hardware*
1	FISICO	*Conversione dei bit in un pacchetto*

Modello TCP/IP

Sono fondamentali per navigare in rete. Tramite i protocolli, i pacchetti di dati vengono trasferiti nella LAN o WAN, quindi anche in Internet. I protocolli del modello TCP/IP funzionano in modo indipendente dall'hardware e dal software sottostante

APPLICAZIONE	*Protocolli http e FTP*
TRASPORTO	*Usa protocolli UDP o TCP*
RETE	*Indirizzamento, routing, controllo del traffico*
INTERFACCIA ALLA RETE	*Nessun protocollo*

Buffer: area della ram per la prelevazione dei pacchetti. Si trovano sia nei dispositivi intermedi, sia in quelli finali. Esso si riempie e se è lento a svuotarsi, perde le informazioni (overflow).

Web Classico (o statico, 1.0)

Il client invia un http request (solo url) al server per richiedere una pagina web al server. Il server risponde con un http response, ovvero invia la pagina web al client, dove nel corpo sarà presente il codice html.

Se c'è un altro client che richiede una pagina, il server manda la stessa pagina.

Client: è il browser (chrome, opera, edge, firefox).

Web Server: Apache, è il più utilizzato.

Il web classico non si può usare ad esempio in un e-commerce perché sennò ognuno avrebbe un carrello "condiviso".

Web dinamico (2.0)

Per risolvere i problemi del web classico, è stato introdotto il web 2.0, a partire dal 2005.

La richiesta del client, oltre l'indirizzo, contiene anche dei dati (come dei cookies) inseriti dall'utente. Questo permette la personalizzazione della risposta da parte del server.

Il client invia un http request, composta dall'url e dai dati inseriti dall'utente. Il web server elabora la risposta ed effettua uno script lato server (php), ovvero la personalizzazione della pagina richiesta dal client. Il server invia al client un http response, ovvero una pagina html personalizzata univoca per ogni richiesta. A questo punto il client può fare uno script lato client con JS per modificare la visualizzazione della pagina (ad esempio, la dark mode)

In questo modo, se due client richiedono la stessa pagina al server, esso restituirà due pagine diverse ai client. Questo metodo è adatto per gli e-commerce.

XAMPP (universale)

È una distribuzione Apache gratuita per creare un server web multipiattaforma tramite tecnologie open source. È composto da Apache, Maria DB, PHP e Perl. Viene utilizzato per creare un web hosting locale sul proprio computer. La X all'inizio sta a significare che è multipiattaforma.

WAMP (solo per windows)

Indica uno stack Windows usato per trasformare il computer in un server locale che ospiti pagine web. Si tratta di una variante dello stack LAMP per sistemi operativi Windows. Le lettere WAMP fanno riferimento al sistema operativo Windows, al server Apache (per eseguire il server web su windows), al DBMS MySQL e al linguaggio PHP, che insieme permettono di costruire una pagina web dinamica. Viene utilizzato per attività di testing: infatti grazie alla configurazione WAMP uno sviluppatore può testare le pagine web sul browser in locale.

LAMP (solo per Linux)

È la stessa cosa di WAMP, però è usato su Linux.