DEFINIZIONE SISTEMI DISTRIBUITI
Un sistema informatico si dice distribuito se una delle due condizioni è verificata:

- **Elaborazione distribuita**: applicazioni risiedono su più host che collaborano tra loro
- **Base di dati distribuita**: patrimonio informativo ospitato su più host

Un sistema distribuito è costituito da un **insieme di applicazioni logicamente indipendenti che collaborano per raggiungere degli obiettivi comuni attraverso un'infrastruttura di comunicazione hardware e software.**

CLASSIFICAZIONE
3 grandi famiglie:

1. Sistemi di calcolo distribuiti: per il calcolo ad alte prestazioni (cluster - grid computing)
2. Sistemi informativi distribuiti: integrano sistemi legacy (presenti in azienda) con nuove tecnologie di comunicazione
3. Sistemi distribuiti pervasivi: connessioni di rete wireless

VANTAGGI
Affidabilità
Dato che sono composti da più nodi (spesso ridondanti) possono continuare a funzionare anche se uno dei componenti si guasta. Tutto ciò grazie alla ridondanza intrinseca, ovvero quando un nodo smette di funzionare, altri nodi possono assumere i suoi compiti. Per ottenere questo risultato è necessario avere strumenti che rilevano automaticamente i guasti e algoritmi che permettano la sostituzione dei nodi,
I sistemi distribuiti possono essere divisi in 2 categorie:

1. Debole accoppiamento: risorse coordinate da un'unica organizzazione
2. Forte accoppiamento: risorse appartengono a organizzazioni diverse che collaborano per fornire servizi

Integrazione
È la capacità di integrare componenti eterogenei (= diversi sia per hardware sia per sistema operativo). Un aspetto importante è la possibilità di collegare sistemi moderni con sistemi legacy.

Trasparenza
È la capacità di un sistema distribuito di apparire come un unico sistema, nascondendo all'utente la complessità interna. Nel modello ISO 10746 esistono 8 forme di trasparenza:

- di accesso: l'utente deve poter utilizzare risorse locali e remote allo stesso modo, senza preoccuparsi di come esse siano fisicamente raggiunte
- di locazione: permette di non mostrare all'utente dove si trova esattamente una risorsa
- di concorrenza: garantisce che più processi possano accedere alla stessa risorsa contemporaneamente
- di replicazione: permette al sistema di gestire automaticamente la duplicazione delle risorse senza che l'utente se ne accorga

- ai guasti: maschera all'utente il verificarsi di un guasto e le eventuali operazioni di recupero
- alla migrazione: nasconde gli spostamenti (fisici o logici) delle risorse
- delle prestazioni: nasconde tutte le operazioni necessarie a riconfigurare il sistema quando cambia il carico di lavoro
- di scalabilità: permette al sistema di crescere senza interrompere il funzionamento

Economicità
I sistemi distribuiti risultano generalmente più economici dei sistemi centralizzati basato su mainframe.

Apertura
Grazie all'uso di diversi standard, i sistemi distribuiti favoriscono l'apertura verso hardware e software di fornitori diversi. Permette a sistemi diversi di cooperare tra loro e ad applicazioni di funzionare su più sistemi operativi senza modificare l'interfaccia.

Connettività e collaborazione
La condivisione delle risorse riduce i costi e favorisce la collaborazione tra utenti.

Prestazioni e scalabilità
Un sistema distribuito può crescere nel tempo aggiungendo nuovi componenti (scalabilità orizzontale). L'aumento delle risorse permette di migliorare le prestazioni e sostenere un maggior numero di richieste. Non esiste un limite definito alle risorse.

Tolleranza ai guasti
Se un componente si rompe, il sistema non si blocca completamente ma subisce al massimo un rallentamento.

SVANTAGGI
Produzione di software
Lo sviluppo del software per sistemi distribuiti è più complesso rispetto alla programmazione tradizionale. I programmatori hanno dovuto adattarsi completamente. L'evoluzione è avvenuta in 3 fasi:

- definizione dello standard TCP/IP, che ha introdotto le socket come base per la comunicazione in rete
- architetture Web, che hanno rivoluzionato la produzione del software, permettendo di creare applicazioni più complesse e interattive.
- linguaggio Java, che permette ai programmi di essere eseguiti su piattaforme diverse senza modifiche.

Complessità
Gestire e prevedere il comportamento complessivo del sistema è molto più complicato.

Sicurezza
L'aumento del numero di nodi connessi comporta anche un aumento delle vulnerabilità. Questo rende necessario adottare strategie di sicurezza molto più avanzate, che garantiscano riservatezza e integrità.

Comunicazione
L'aumento continuo del numero di utenti provoca una costante richiesta di maggiore larghezza di banda e migliori prestazioni.
Il trasferimento delle informazioni a distanza richiede infrastrutture di telecomunicazione sempre più avanzate.

COMUNICAZIONE IN UN SISTEMA CLIENT SERVER
Il Web è basato sul modello client-server che ha 2 differenti elementi:

- client
- server

I client sono elementi ATTIVI (Web browser) che:

- utilizzano il protocollo HTTP per connettersi ai server
- usano un URL per identificare le risorse
- richiedono pagine Web ai server e ne visualizzano il contenuto.

I server sono elementi PASSIVI (Web server o HTTP server) che:

- rimangono in ascolto di eventuali connessioni di nuovi client su una determinata porta TCP
- utilizzano il protocollo HTTP per interagire con i client
- forniscono ai client le pagine Web o le risorse richieste.

HTTP (Hyper Text Transfer Protocol), è il protocollo principale su cui si basa il World Wide Web (WWW). Viaggia sopra TCP/IP e lato server utilizza la **porta 80**. HTTP è nato dall'insieme di standard HTML e URL/URI:

- **HTML**: linguaggio di markup per le pagine web.
- **URL/URI**: specifica gli indirizzi delle risorse da visualizzare. Include informazioni come:
    - Nome dell'host
    - Percorso della risorsa
    - Nome del file
    - Eventuali query o tag all'interno della pagina

La comunicazione tramite HTTP avviene all'interno di **sessioni**, che iniziano sempre con l'apertura di una connessione TCP verso il server. Una volta stabilita la connessione, il client invia una **request**, contenente tra l'altro l'URL della risorsa richiesta. Il server, elabora la richiesta, restituisce il file richiesto (**response**) e chiude la connessione TCP.

Riga di richiesta
I metodi principali sono:

- **GET**: richiesta semplice di una pagina o risorsa; i dati vengono inviati in chiaro nella URL, ad esempio `?Nome=valore&nome=valore`.

- **POST**: invio di dati al server; i dati della form vengono inviati nascosti nel body della richiesta. Le form usano tipicamente il metodo POST, perché i dati non sono visibili nella URL.
- Entrambi i metodi possono essere usati sia per inviare dati sia per riceverli.

Comunicazione unicast e multicast

Nel caso della comunicazione **unicast**, il server rimane in ascolto sulla porta 80, pronto a ricevere connessioni in arrivo dai client. Quando un client tenta di stabilire una connessione, server e client si accordano su una porta libera superiore alla 5000 per continuare la comunicazione. La porta 80 rimane comunque libera per accettare nuove connessioni. Una volta stabilita la connessione, il server crea un thread figlio dedicato a gestire la comunicazione con quel particolare client, lasciando a questo thread tutta la responsabilità della gestione della connessione, mentre il server principale continua a monitorare eventuali nuove richieste.

La comunicazione **multicast**, invece, si distingue per il fatto che il destinatario non è un singolo client, ma più client contemporaneamente. In questo caso, il server invia periodicamente informazioni a tutti i client della rete, ad esempio dati provenienti dai sensori di temperatura. Ogni client riceve le informazioni e decide autonomamente se eseguire le istruzioni ricevute. A differenza della comunicazione unicast, in multicast il server non riceve richieste dai client, ma agisce inviando messaggi a tutti senza attendere una richiesta specifica.

SOCKET
COSA SONO
I **socket** sono un meccanismo che permette di identificare univocamente un processo in esecuzione su un host e di stabilire la comunicazione tra processi su host diversi. Non basta conoscere solo l'indirizzo IP di un host, perché su un singolo computer possono girare molti processi contemporaneamente. Per questo motivo, un socket combina due informazioni fondamentali: l'indirizzo IP dell'host e il **numero di porta**, che identifica in modo univoco il servizio o processo specifico su quell'host.
Il server registra ogni servizio disponibile associandolo a un numero di porta tramite il protocollo di trasporto (operazione chiamata **bind**), mentre il client per contattare il servizio deve conoscere sia l'indirizzo IP dell'host sia il numero di porta associato al servizio desiderato. In questo modo, il socket consente la comunicazione precisa tra client e server, indirizzando correttamente i messaggi al processo giusto.

SERVIZI DIFFUSI ASSOCIATI
I servizi più diffusi associati ai socket includono il servizio di posta elettronica, che utilizza il protocollo SMTP sulla porta TCP 25; il servizio web, che utilizza il protocollo HTTP sulla porta TCP 80; e il servizio DNS, che traduce i nomi dei calcolatori in indirizzi IP e risponde sulla porta UDP 53.

FORMATO XML
CARATTERISTICHE
Il formato XML, acronimo di **eXtensible Markup Language**, non è un linguaggio di markup standard né un'evoluzione di HTML. Si tratta di un **metalinguaggio**, cioè un linguaggio che consente di definire altri linguaggi di markup.

DIFFERENZE CON HTML
A differenza di HTML, XML non ha tag predefiniti e non serve né per programmare né per definire pagine web. La sua funzione principale è fornire un insieme standard di regole sintattiche per descrivere la struttura dei dati e dei documenti. Mentre HTML si occupa di **presentare e visualizzare i dati**, XML si concentra su **descrivere la natura dei dati**, consentendo di rappresentarli in modo gerarchico e leggibile dall'uomo. In pratica, XML permette di aggiungere tag ai dati per definire il loro significato, senza specificarne lo stile o il modo in cui devono essere visualizzati.

REGOLE FORMATO XML
Per garantire la correttezza dei documenti XML, esistono regole sintattiche fondamentali. Un documento XML è detto **well-formed** quando rispetta tutte le seguenti regole:

- Deve contenere una dichiarazione XML corretta all'inizio del documento.
- Deve avere un unico elemento radice, esclusi eventuali commenti.
- Ogni elemento deve avere un tag di apertura e uno di chiusura; se l'elemento è vuoto, è possibile usare la forma abbreviata `<nometag/>`.
- Gli elementi devono essere correttamente nidificati, cioè i tag di chiusura devono seguire l'ordine inverso dei rispettivi tag di apertura.
- I nomi dei tag di apertura e chiusura devono coincidere, rispettando maiuscole e minuscole.
- I nomi dei tag non devono iniziare con un numero o con il carattere underscore (_), e non possono contenere spazi al loro interno.
- I valori degli attributi devono sempre essere racchiusi tra apici singoli o doppi.

Un documento XML è definito **valido** quando, oltre a essere well-formed, rispetta anche le regole semantiche definite da uno schema o da un **Document Type Definition (DTD)**. Il DTD specifica quali elementi e attributi possono essere utilizzati e in quale contesto, ma non definisce il tipo di contenuto o i valori precisi degli attributi.

FORMATO JSON
TIPI DI DATI E STRUTTURE

Il formato JSON, acronimo di **JavaScript Object Notation**, è uno standard aperto per la rappresentazione e lo scambio di dati tra applicazioni, sia stand-alone che via Web. JSON è semplice da leggere e scrivere, e rispetto a XML produce documenti più compatti perché privo di schemi e tag di marcatura complessi. La struttura di JSON si basa su due elementi fondamentali: **insiemi di coppie nome/valore** e **liste ordinate di valori**.

JSON supporta diversi tipi di dati:

- i numeri (Number) sono in formato a virgola mobile a doppia precisione;
- le stringhe (String) sono sequenze di caratteri Unicode racchiuse tra virgolette;
- i valori booleani (Boolean) possono essere `true` o `false`;
- gli array (Array) rappresentano sequenze ordinate di valori racchiuse tra parentesi quadre;
- gli oggetti (Object) sono insiemi non ordinati di coppie chiave/valore racchiusi tra parentesi graffe;
- gli spazi bianchi (Whitespace) possono essere inseriti liberamente per migliorare la leggibilità;
- il valore `null` rappresenta un elemento vuoto o non inizializzato.

DIFFERENZE/VANTAGGI JSON vs XML

JSON e XML sono entrambi formati utilizzati per lo scambio di dati, ma presentano differenze significative.

XML è estremamente flessibile e consente di creare strutture gerarchiche complesse con tag personalizzati, il che lo rende adatto a documenti molto articolati. Tuttavia, in progetti grandi può risultare meno immediatamente leggibile e richiede una maggiore attenzione per comprendere l'organizzazione dei dati.

JSON, invece, possiede una struttura più semplice e compatta, che consente di comprendere quasi subito il significato dei dati. La semplicità di JSON lo rende particolarmente utile in applicazioni web collaborative e in progetti in cui programmatori diversi utilizzano linguaggi diversi. Inoltre, JSON si integra facilmente con JavaScript, consentendo di elaborare direttamente le informazioni nel contesto di applicazioni web, cosa che XML non fa senza parser esterni.