### Evoluzione dei sistemi informativi

Nel tempo, i sistemi informativi sono passati da modelli centralizzati a modelli distribuiti.

- Nei sistemi centralizzati, tutte le applicazioni e i dati si trovano su un unico elaboratore.
- Nei sistemi distribuiti, invece, le applicazioni sono suddivise su più nodi che cooperano tra loro per raggiungere obiettivi comuni.

---

### Tipologie di applicazioni in un sistema distribuito

A seconda del ruolo svolto:

- **Client**: utilizza i servizi offerti da altri
- **Server**: fornisce servizi ad altri
- **Actor**: svolge sia il ruolo di client che di server

---

### Classificazione dei sistemi distribuiti

1. **Sistemi di calcolo**
   - Cluster computing: insieme di PC omogenei connessi tramite rete locale
   - Grid computing: insieme di PC eterogenei, distribuiti
2. **Sistemi informativi distribuiti**: come il Web, i sistemi aziendali
3. **Sistemi pervasivi**: dispositivi mobili, reti domestiche, wearable, reti di sensori

---

### Vantaggi principali dei sistemi distribuiti

- **Affidabilità**: in caso di guasti, il sistema continua a funzionare grazie alla ridondanza
- **Integrazione**: supporta componenti eterogenei (hardware e software)
- **Trasparenza**: l'utente percepisce il sistema come un unico elaboratore

Le otto forme di trasparenza secondo l'ISO 10746:

1. **Access transparency**: permette di accedere alle risorse allo stesso modo, sia locali che remote
2. **Location transparency**: l'utente non deve conoscere la posizione della risorsa
3. **Concurrency transparency**: più utenti possono accedere contemporaneamente senza conflitti
4. **Replication transparency**: le risorse possono essere duplicate senza che l'utente se ne accorga
5. **Failure transparency**: i guasti sono mascherati e gestiti automaticamente
6. **Mobility transparency**: il movimento delle risorse non cambia il modo di accedervi
7. **Performance transparency**: le modifiche per ottimizzare le prestazioni sono invisibili all'utente
8. **Scaling transparency**: il sistema può essere ampliato senza influire sugli utenti

Altri vantaggi:

- **Economicità**: costi più contenuti rispetto ai sistemi centralizzati

- **Apertura**: uso di protocolli standard, che favoriscono l'interoperabilità
- **Collaborazione**: condivisione efficiente di risorse
- **Scalabilità**: il sistema cresce facilmente in base alle esigenze
- **Tolleranza ai guasti**: anche in caso di errori, il sistema continua a operare

---

#### Svantaggi dei sistemi distribuiti

- **Complessità**: maggiore rispetto ai sistemi centralizzati, richiede strumenti e competenze specifiche
- **Sicurezza**: maggiore esposizione a rischi come sniffing e spoofing
- **Comunicazione**: richiede reti avanzate e affidabili
- **Difficoltà nello sviluppo software**: servono strumenti specifici e conoscenze distribuite

---

#### Architetture hardware

Classificazione di Flynn (1972):

- **SISD**: un solo flusso di dati e istruzioni (es. PC, mainframe)
- **SIMD**: un solo flusso di istruzioni su più flussi di dati (es. vector processor)
- **MISD**: più flussi di istruzioni su un unico flusso di dati (raro)
- **MIMD**: più flussi di istruzioni su più flussi di dati, con:
  - memoria condivisa (multiprocessor)
  - memoria privata (multicomputer, comunicazione tramite messaggi)

##### Cluster e Grid

- **Cluster**: nodo centrale, alta velocità, tutti i computer sono fisicamente vicini
- **Grid**: maggiore eterogeneità e distribuzione geografica, sicurezza e policy differenziate

##### Sistemi pervasivi

- Nodi piccoli, spesso mobili, connessioni wireless
- Si autoconfigurano, non richiedono competenze tecniche da parte dell'utente

---

#### Architetture software

1. Terminali remoti: terminali senza capacità di elaborazione
2. Client-server: struttura base di richiesta e risposta
3. Web-centric: server centrale che gestisce tutti i servizi
4. Cooperative: entità autonome che collaborano
5. Completamente distribuite: entità paritarie senza un server centrale
6. A livelli: ogni componente ha una responsabilità specifica (tipico dei sistemi moderni)

---

#### Comunicazione web e HTTP

- Protocollo HTTP client-server su porta 80
- Si basa su TCP
- La connessione può essere:
  - Incanalata: una richiesta alla volta
  - Non incanalata: più richieste in coda

**Struttura dei messaggi HTTP**:

- Request: metodo, URI, versione, intestazioni, corpo (opzionale)
- Response: versione, codice di stato, intestazioni, corpo (HTML, JSON...)

**Metodi HTTP più comuni**:

- GET: recupera dati
- POST: invia dati
- PUT: crea o modifica risorse
- DELETE: elimina risorse

**Codici di stato**:

- 1xx: informazioni
- 2xx: successo
- 3xx: redirezione
- 4xx: errore del client
- 5xx: errore del server

**Codifica URL**:

- I caratteri speciali vengono trasformati in formato `%XX` dove XX è il codice ASCII
- Spazi diventano `+`
- Nomi e valori sono separati da `=` e uniti da `&`

Applicazioni Web

- **Tecnologie client-side**: visibili, eseguite dal browser (es. HTML, CSS, JavaScript)
- **Tecnologie server-side**: non visibili, eseguite sul server (es. PHP, Java)

**Linguaggi**:

- Markup: HTML
- Programmazione: Java, JS, PHP

Architettura client-server

- Il client invia una richiesta, il server la riceve e risponde
- I server restano in ascolto su una specifica porta (socket = IP + porta)
- Due tipi di comunicazione:
  - Unicast: un client alla volta
  - Multicast: più client in parallelo con gestione tramite thread separati

### Architettura a livelli (3-tier)

- Presentation Layer (interfaccia utente)
- Business Logic Layer (logica dell'applicazione)
- Data Access Layer (accesso ai dati)

Thin client: logica sul server
Thick client: logica anche sul client

### Applicazioni di rete

- I protocolli di rete vengono gestiti nel livello applicazione del modello TCP/IP

**Protocolli principali**:

- HTTP: web
- FTP: trasferimento file
- SMTP: invio mail
- POP3 / IMAP4: ricezione mail
- DNS: risoluzione nomi
- SNMP: gestione reti

### Socket

- Combinazione di IP e porta
- Ogni connessione è gestita da un thread separato che utilizza un socket dinamico
- Il socket iniziale (socket di benvenuto) accetta nuove connessioni

### Architetture di rete

- **Client-Server**: server con IP fisso, può usare server farm per distribuire il carico
- **P2P (peer-to-peer)**: ogni nodo è sia client che server, IP dinamici, decentralizzazione totale
- **P2P centralizzato**: server centrale che conserva gli indici, ma i file risiedono sui peer
- **P2P ibrido**: super peer (supernodi) per indicizzazione, altri peer semplici