**TECNOLOGIE A LIVELLO DI ARCHITETTURA FISICA**

**Classificazione delle architetture fisiche**

Le architetture fisiche possono essere innanzitutto *centralizzate* o *distribuite*.

I SI *centralizzati* hanno un solo nodo di elaborazione in cui è installato l’intero software necessario e vari terminali capaci di comunicare con il nodo centrale attraverso periferiche di input e di output (tastiere, schermi, …). I terminali possono interagire con il nodo centrale, ma non eseguire le applicazioni in locale.

I SI *distribuiti*, invece, per essere definiti tali devono avere almeno una delle seguenti caratteristiche:

* Le varie applicazioni del sistema devono risiedere su diversi nodi di elaborazione;
* Il patrimonio informativo deve risiedere su diversi nodi di elaborazione;

Quindi un SI distribuito ha vari nodi di elaborazione, che possono eseguire applicazioni in locale, e può avere più applicazioni indipendenti che collaborano.

**Layer Applicativi**

È possibile descrivere le applicazioni dal punto di vista dell’architettura fisica secondo tre livelli logici, detti *layer*, suddivisi così:

1. Livello di *presentazione* (P): si occupa di gestire le interazioni con l’utente, come l’interfacciamento grafico e il rendering delle informazioni. È detto anche livello front-end dell’applicazione;
2. Livello della *logica applicativa* (A): si occupa delle *funzioni* che vengono messe a disposizione all’utente;
3. Livello di logica *di accesso ai dati* (D): si occupa della gestione dell’informazione, con accesso a database o a risorse disponibili online.

I livello di logica applicativa e di accesso ai dati sono back-end e si può accedere ad esse attraverso delle API.

**Livelli hardware - Tier**I livelli logici (layer) vengono installati su livelli hardware, detti *tier*. Ogni livello hardware corrisponde a una macchina o a un insieme di macchine.

Esistono diversi tipi di architetture a seconda di quanti livelli hardware l’applicazione impiega.

La più semplice è la *single tiered*, cioè quella in cui tutti i layer logici sono installati su un’unica macchina. Essa corrisponde a un’architettura di tipo *centralizzato*, rifacendoci alla categorizzazione vista in precedenza.

Un’altra architettura è la *two tiered*, basata su un’architettura di tipo client-server. Il client può essere di tipo *thin* se su di esso è implementato solo il livello logico P, o thick se oltre al livello P sul client è implementato anche il livello A.

Una terza architettura è la *three tiered*, basata su tre livelli di distribuzione. Questi tre livelli hardware corrispondono grossomodo ai tre livelli logici. Quindi i tre layer sono distribuiti in linea di principio su altrettante macchina dedicate. L’approccio three tiered è quello più flessibile e più scalabile. Un esempio di architettura three tiered prevede un server DBMS (layer D), un server intermedio che alleggerisce il carico del DBMS gestendo al suo posto la logica applicativa (layer A) e il client ha l’interfaccia che permette l’interazione con l’applicazione (livello P).

Esiste anche un tipo di architettura *n-tiered*, che distribuisce i layer applicativi su un generico numero n di tier. In questa architettura ogni funzionalità ha una macchina dedicata.

**Scalabilità**

La scalabilità di un’architettura è la capacità dell’infrastruttura di soddisfare richieste crescenti.

Esistono due tipi di scalabilità:

* Scalabilità *verticale*: non si aumenta il numero di nodi (macchine), ma si aumentano le loro prestazioni, aggiungendo a quelle esistenti più CPU o RAM ad esempio.
* Scalabilità *orizzontale*: si individua il nodo più lento e lo si sostituisce con più nodi dello stesso tipo che si dividono il suo carico.

Quando si devono aumentare le prestazioni del sistema, è molto importante scegliere correttamente *quanto* aumentarle (ad esempio quanta RAM aggiungere o quanti nodi in più inserire) : bisogna aumentare le prestazioni abbastanza da sopportare l’aumento di richieste da parte degli utenti, ma non bisogna aumentarle in modo eccessivo, o il costo dell’operazione andrà a superare i benefici che derivano dall’ampliamento.

In ambito web, la tendenza è quella di avere sistemi che sappiano essere *elastici* piuttosto che scalabili, cioè che possano aumentare le prestazioni quando aumenta il carico di lavoro e diminuirle quando diminuisce il carico.

Un esempio di approccio che favorisce molto la scalabilità sono le server farm, cioè insieme di elaboratori che si dividono il carico elaborativo.

Le server farm possono seguire due principi progettuali diversi: la clonazione e il partizionamento. Nell’approccio a clonazione sui server sono installati gli stessi software e forniti gli stessi dati, quindi ogni “clone” svolge le stesse operazioni e i vari server si dividono il carico di lavoro in modo equo. I dati possono essere forniti come copia (completa, contenente tutti i dati) a sé stante per ogni server, oppure tutti i server possono accedere ai dati attraverso un unico server che memorizza un’unica copia dei dati e li fornisce a tutti i cloni. Un insieme di cloni dedicati a un particolare servizio è detto RACS.

L’approccio a partizionamento invece prevede che ogni nodo svolga un’azione specializzata. I vari nodi hanno lo stesso hardware e software, ma non gli stessi dati, che in questa configurazione vengono ripartiti fra i vari nodi. Per evitare che in caso di guasto di uno dei nodi un’intera funzionalità non sia più disponibile, si effettua una clonazione anche dei nodi del sistema partizionato, così che esistano almeno due nodi che abbiano accesso a una specifica partizione di dati e quei dati non diventino inaccessibili in caso di guasto. Questo approccio è detto RAPS.

Virtualizzazione

Un modo di gestire le architetture fisiche è la virtualizzazione. La virtualizzazione consiste nell’usare componenti software che replichino le componenti hardware, ad esempio CPU virtuali al posto delle CPU fisiche. La CPU virtuale viene invocata tramite applicazione, e “gira” la chiamata alla CPU fisica ad essa associata.

L’insieme di risorse virtuali necessarie al funzionamento di un’applicazione è detta macchina virtuale.

La corrispondenza fra risorse fisiche e virtuali è gestita da un hypervisor (ad esempion VirtualBox), che rende possibile l’esecuzione della macchina virtuale. Su una sola macchina fisica si possono installare più macchine virtuali.

**Cloud computing**

Il cloud computing è un paradigma che permette creare architetture applicative che, basandosi su sistemi virtualizzati, possono accedere a certe risorse da qualunque dispositivo che sia connesso alla rete.

Le principali caratteristiche del cloud computig sono:

* on demand self service: gli utenti ottengono in modo autonomo risorse, ad esempio tempo di esecuzione o spazi di memorizzazione;
* accesso alla rete onnipresente: si può accedere alle risorse cloud con qualunque dispositivo abbia accesso internet;
* accesso alle risorse indipendente dalla localizzazione: non importa su che server le risorse siano memorizzate;
* elasicity: le risorse possono cambiare di numero molto rapidamente su richiesta dell’utente;
* monitoraggio sull’utilizzo: il costo delle risorse cloud è calcolato sull’uso effettivo che se ne fa.

I modelli di servizio basati sul cloud computing sono:

* Infrastructure as a Service (Iaas): il cloud provider mette a disposizione risorse di tipo fisico (server, storage , …). Queste possono ospitare le applicazioni degli utenti che sono installate su macchine virtuali gestite dagli utenti stessi.
* Platform as a Service (Paas): il cloud fornisce mette a disposizione tecnologie a livello di piattaforma, ad esempio servizi di DBMS. L’utente accede al servizio tramite API.
* Software as a Service (SaaS) : il cloud provider fornisce, oltre a risorse PaaS e IaaS, anche applicazioni in cloud. L’utente non può modificare attivamente il servizio fornito, ma può configurare le applicazioni attraverso alcuni parametri (Ad esempio Google Drive, Dropbox).
* Everything as a Service (\*aaS): altri elementi vengono forniti come servizi dal cloud provider, ad esempio dati, processi di business, …

Il cloud computer può presentarsi in quattro diversi modelli di deployment:

* Private cloud: l’infrastruttura è fornita a uso esclusivo di una singola organizzazione.
* Community cloud: l’infrastruttura è fornita ad uso di una comunità di organizzazioni con un obiettivo comune.
* Public cloud: l’infrastruttura è fornita ad uso pubblico
* Hybrid cloud: l’infrastruttura si compone di due o più infrastrutture dei modelli precedenti, che collaborano ma rimangono due entità distinte.

**Opzioni di gestione del SI**

Si può gestire un SI in-house, quindi all’interno dell’organizzazione a cui appartiene, oppure in outsourcing, affidandolo a un’organizzazione esterna. Si possono definire dei livelli a seconda del grado di outsourcing:

* Livello 0: tutto il SI è gestito in-house, dall’azienda stessa;
* Livello 1: l’organizzazione è supportato da un’azienda IT che lavora in modo esclusivo per l’organizzazione stessa;
* Livello 2: l’organizzazione gestisce il SI in-house ma con il supporto di un’organizzazione esterna;
* Livello 3: alcune risorse sono gestite in out-sourcing, ad esempio possono essere condivise da più organizzazioni (ad esempio un data center che contiene i dati di diverse organizzazioni);
* Livello 4: oursourcing selettivo, alcune componenti sono affidate ad aziende esterne;
* Livello 5: Outsourcing completo, tutto il SI è gestito all’esterno dell’organizzazione.

Le organizzazioni che forniscono servizi di outsourcing sono dette service provider e possono fare supporti di housing, cioè forniscono all’utente uno spazio fisico dove mettere i server, oppure hosting, cioè forniscono servizi in rete.