Architetture e Infrastrutture Hardware/Software - Guida Completa

1. CONCETTI FONDAMENTALI

1.1 Definizioni Base

Sistema Informatico: È l'insieme di tutti i componenti hardware e software che collaborano tra loro per elaborare, memorizzare e trasmettere informazioni. Come un'orchestra sinfonica, ogni componente ha il suo ruolo specifico, ma è la coordinazione generale che crea l'armonia finale.

Architettura vs Infrastruttura:

- **Architettura**: È come progettare una casa - decidi dove mettere le stanze, come collegarle, quale sarà il flusso delle persone. È la progettazione logica del sistema.
- **Infrastruttura**: Sono i mattoni, il cemento, gli impianti elettrici e idraulici - tutti i componenti fisici e virtuali che permettono alla casa di funzionare.

Esempio pratico con Netflix:

- *Architettura*: Come i microservizi comunicano tra loro, come è organizzato il sistema di raccomandazioni
- *Infrastruttura*: I server AWS su cui gira, la rete globale di distribuzione, i database

1.2 Architettura Software

L'**architettura software** rappresenta il sistema operativo e l'ambiente dove vengono eseguiti vari oggetti software, ciascuno con uno scopo specifico. Combina quattro elementi fondamentali:

1. **Struttura**: Gli stili architetturali utilizzati
2. **Caratteristiche architetturali**: I requisiti di qualità del sistema
3. **Decisioni architetturali**: I vincoli e le regole inviolabili
4. **Principi di design**: Le linee guida per le scelte quotidiane

2. ARCHITETTURA HARDWARE

2.1 Componenti Fondamentali

CPU (Central Processing Unit)

- È il "cervello" del computer che esegue le istruzioni
- Come un direttore d'orchestra che coordina tutti i musicisti
- Nei server: più core, più thread, cache più grande (es. Intel XEON)

- Si collegano ai socket sulla motherboard

Memoria RAM

- Come la scrivania di uno studente: tiene a portata di mano tutto quello che sta usando
- Più grande è la RAM, più programmi possono essere aperti contemporaneamente
- È volatile: si cancella allo spegnimento
- **RAM ECC nei server**: Rileva e corregge automaticamente errori nei dati

Storage (Memoria di Massa)

- Come un magazzino per conservare tutto a lungo termine
- **HDD**: Come magazzini tradizionali (più lenti ma capienti)
- **SSD**: Come magazzini automatizzati moderni (velocissimi)
- **RAID**: Combina più dischi per prestazioni/ridondanza

Bus e Interconnessioni

- Le "strade" del computer per la comunicazione tra componenti
- **PCIe**: Standard ad alta velocità per schede video, rete, storage
- **DMI**: Comunicazione tra processore e chipset
- **Chipset**: Gestisce comunicazione tra CPU, memoria e periferiche

2.2 Tipologie di Architettura

Architettura Von Neumann

- Come avere un'unica biblioteca per libri di testo (dati) e manuali (programmi)
- Semplice ma può creare "traffico" quando CPU accede a dati e istruzioni

Architettura Harvard

- Come avere due biblioteche separate per dati e istruzioni
- Elimina il traffico ma è più complessa

Architetture Multicore

- Come avere più chef in cucina: ognuno prepara un piatto diverso contemporaneamente

2.3 Gerarchia della Memoria

1. **Cache CPU**: Dati più utilizzati, velocissima ma piccola
2. **RAM**: Dati dei processi attivi, veloce ma volatile
3. **Storage**: Tutto il resto, lenta ma persistente

Collo di bottiglia: Quando un processo richiede più memoria di quella disponibile.

3. ARCHITETTURA SOFTWARE

3.1 Caratteristiche di una Buona Architettura

Sicurezza: Come una banca con sistemi di allarme, casseforti e controlli di accesso

Affidabilità: Come un'auto che anche se si rompe una ruota, ha la ruota di scorta e può continuare (tolleranza ai guasti)

Scalabilità: Come un ristorante che può facilmente aggiungere tavoli quando arrivano più clienti

- **Verticale (Scale-up)**: Aggiungere potenza a un nodo esistente
- **Orizzontale (Scale-out)**: Aggiungere nuovi nodi

Manutenibilità: Come un motore ben progettato, facile da riparare e aggiornare

Evitare Single Point of Failure: Seguire pattern di sviluppo solidi e ridondanti

3.2 Pattern Architetturali

Monolitico

- Come un palazzo dove tutto è un unico grande appartamento
- Tutte le funzioni in un singolo blocco eseguibile
- Facile da sviluppare inizialmente, difficile da scalare

Client-Server

- Come un ristorante: cliente ordina, server elabora e risponde
- Molto usato per applicazioni web

Layered (A Livelli)

- Come un grattacielo organizzato per piani
- Ogni livello dipende solo da quello sottostante
- Esempio: Stack TCP/IP
- Garantisce semplicità, sicurezza e scalabilità

Microservizi

- Come una food court: ogni servizio fa una cosa specifica
- Applicazione divisa in servizi indipendenti che comunicano
- Aumenta scalabilità ma anche complessità
- Esempi: Netflix, Amazon

Event-Driven

- Come sistema di notifiche push
- I componenti reagiscono a eventi generati da altri componenti

3.3 Il Ruolo dell'Architetto Software

L'architetto software è come un diplomatico che deve mediare tra mondi diversi:

Competenze Chiave:

- Guidare scelte tecnologiche con giustificazioni
- Bilanciare profondità e ampiezza (vs sviluppatore che è specializzato)
- Navigare la politica aziendale
- Mantenere le mani sporche di codice

Differenza con Sviluppatore:

- Sviluppatore = chirurgo specializzato (grande profondità)
- Architetto = medico di famiglia (grande ampiezza)

4. INFRASTRUTTURE

4.1 Infrastruttura On-Premise

Server Fisici

- Come avere un datacenter nel tuo palazzo
- Controllo totale ma devi gestire tutto tu
- Elementi ridondanti (alimentatori, ventole, dischi)
- Progettati per operare 24/7

Componenti Server:

- **Motherboard specializzate**: Multi-socket, ECC RAM, più slot PCIe
- **Sistemi di raffreddamento**: Flusso front-to-back, collettori aria calda
- **Alimentatori ridondanti**: Resistenti a sbalzi di corrente
- **Schede di rete**: NIC specializzate per traffico intenso

Formati Server:

- **Rack**: Misurati in unità (U)
- **Blade**: Moduli in chassis condiviso

4.2 Virtualizzazione

Virtualizzazione Tradizionale

- Come dividere un appartamento in appartamentini indipendenti
- Ogni VM ha il suo sistema operativo completo
- **Hypervisor Tipo 1**: Installato direttamente sull'hardware (migliori prestazioni)
- **Hypervisor Tipo 2**: Come applicazione su SO host (es. VirtualBox)

Containerizzazione (Docker)

- Come stanze in hotel: condividono servizi comuni ma spazi privati
- Più efficiente: condividono kernel del SO host
- Risolve problema "funziona sulla mia macchina"
- Tecnologia disruptive per deployment

4.3 Tecnologie di Storage

NAS (Network Attached Storage)

- Dispositivo storage collegato alla rete IP
- Fornisce file sharing centralizzato
- Vantaggi: Semplice, economico, centralizzato
- Svantaggi: Limitato dalla rete, non adatto per accesso raw

SAN (Storage Area Network)

- Rete dedicata allo storage, separata dalla LAN
- Accesso a livello di blocco (raw)
- **Fiber Channel**: Alte prestazioni, costi elevati
- **iSCSI**: Usa protocollo IP, più economico
- **FCoE**: Vantaggi FC su Ethernet

RAID (Redundant Array of Independent Disks)

- **RAID 0**: Striping - prestazioni elevate, nessuna ridondanza
- **RAID 1**: Mirroring - dati duplicati, alta protezione
- **RAID 5+**: Combinazioni di prestazioni e ridondanza

4.4 Cloud Computing

IaaS (Infrastructure as a Service)

- Come affittare casa vuota: ti danno l'immobile, tu porti i mobili
- Esempi: AWS EC2, Azure VMs

PaaS (Platform as a Service)

- Come affittare casa ammobiliata: ambiente di sviluppo incluso
- Esempi: Heroku, Google App Engine

SaaS (Software as a Service)

- Come stare in hotel: tutto pronto, devi solo usarlo
- Esempi: Gmail, Office 365, Netflix

Vantaggi Cloud:

- Scalabilità elastica
- Pay-per-use (paghi solo quello che usi)
- Agilità (servizi attivi in minuti)

Svantaggi Cloud:

- Dipendenza dalla rete
- Questioni di sicurezza e conformità
- Vendor lock-in

5. ALTA DISPONIBILITÀ E SCALABILITÀ

5.1 High Availability (HA)

Ridondanza: Come avere più autisti per un taxi - se uno si ammala, gli altri continuano

Failover: Come generatore di emergenza - se va via corrente, backup si attiva automaticamente

Load Balancing: Come dispatcher che manda clienti alla cassa meno affollata

SLA (Service Level Agreement): Definisce disponibilità richiesta (es. 99.9% = 8 ore downtime/anno)

5.2 Strategie di Scalabilità

Scaling Verticale:

- Aggiungere risorse a nodo esistente
- Come comprare computer più potente
- Semplice ma costoso e limitato

Scaling Orizzontale:

- Aggiungere nuovi nodi
- Come assumere più camerieri
- Più complesso ma flessibile

6. MISURAZIONE E QUALITÀ

6.1 Metriche di Complessità

Complessità Ciclomatica

- Misura "bivi" decisionali nel codice (if, while, switch)
- Come contare incroci in città: più incroci = più difficile navigare
- Regola: sotto 10 accettabile, sotto 5 preferibile

Modularità

- **Coesione**: Quanto elementi di modulo "appartengono" insieme
- **Coupling**: Quanto moduli dipendono l'uno dall'altro
- **Connascence**: Classificazione sofisticata delle dipendenze

6.2 Fitness Functions

Controlli automatici di qualità architetturale

- Come controlli qualità in fabbrica auto
- Verificano continuamente rispetto dei principi
- Consulente architetturale 24/7

7. SICUREZZA E MONITORAGGIO

7.1 Sicurezza a Livelli (Defense in Depth)

Perimetro: Firewall come guardie all'ingresso **Accesso**: Autenticazione come badge personali **Applicazione**: Protezione SQL injection, input validation **Dati**: Crittografia come casseforti

7.2 Monitoraggio

Logging: Diario di bordo che registra tutto **APM (Application Performance Monitoring)**: Sensori vitali del sistema **Alerting**: Allarmi che avvisano quando qualcosa va storto

8. TECNOLOGIE DISRUPTIVE

8.1 Evoluzione Tecnologica

PC vs Mainframe: Democratizzazione del computing **Internet**: Trasformazione comunicazione e business **Cloud**: Cambio paradigma IT infrastructure
Docker: Rivoluzione deployment e portabilità **AI/LLM**: Automazione intelligente e generazione contenuti

8.2 Caratteristiche Tecnologie Disruptive

- Cambiano paradigma operativo
- Spesso basate su tecnologie esistenti da tempo
- Creano nuovi modelli di business
- Rendono obsolete tecnologie precedenti

9. STRATEGIE DI PROGETTAZIONE

9.1 Principi Guida

Non cercare architettura perfetta, ma meno peggiore

- Ogni scelta ha trade-off
- Come scegliere dove vivere: centro (comodo/costoso) vs periferia (economico/scomodo)

Progettare per iterazione

- Sistema deve poter evolvere
- Fondamenta che supportano piani aggiuntivi

Massimo 3 caratteristiche principali

- Non puoi ottimizzare tutto
- Come auto: veloce, economica, affidabile - scegli 2 su 3

9.2 Identificazione Componenti

Approccio Actor/Actions: Chi usa sistema e cosa fa **Event Storming**: Eventi importanti nel sistema
Workflow Approach: Seguire flusso di lavoro end-to-end

10. SERVER E HARDWARE AVANZATO

10.1 Componenti Server Specializzati

Differenze Server vs Desktop:

- CPU multi-socket (Xeon/EPYC vs Core/Ryzen)
- RAM ECC vs non-ECC
- Più slot PCIe, RAID avanzato

- Alimentatori ridondanti
- Form factor Rackmount/Blade

10.2 Networking e Gestione

KVM Switch: Controllo multipli server con singolo set tastiera/video/mouse **IPMI/iLO**: Gestione remota server **Schede di rete specializzate**: Per traffico intenso

10.3 Raffreddamento Avanzato

Sistemi tradizionali: Ventole, dissipatori, liquido **Immersion cooling**: Server immersi in liquido dielettrico **Progettazione datacenter**: Corsie calde/fredde, contenimento

11. ARCHITETTURE COMPLESSE

11.1 Blade Server

Chassis condiviso: Alimentazione, raffreddamento, rete **Moduli specializzati**: CPU, storage, rete **Vantaggi**: Alta densità, cablaggio ridotto, manutenzione centralizzata **Svantaggi**: Costi elevati, vendor lock-in

11.2 Virtualizzazione Avanzata

Linked Clone: VM condivide disco base, risparmia spazio **Full Clone**: Copia indipendente, più stabile ma occupa più spazio **VDI (Virtual Desktop Infrastructure)**: Desktop virtuali centralizzati

11.3 Tipi di Virtualizzazione

Server: Hypervisor tipo 1 (bare metal) vs tipo 2 (hosted) **Rete**: SDN (Software Defined Networking), NFV (Network Function Virtualization) **Applicazioni**: Separazione app da SO, miglior compatibilità **Storage**: Astrazione risorse storage fisiche

CONCLUSIONI E CONSIGLI PRATICI

Concetti Chiave da Ricordare:

1. **Sistema = Insieme coordinato** di componenti verso obiettivo comune
2. **Architettura** combina struttura, caratteristiche, decisioni, principi
3. **Architetto** bilancia competenze tecniche, business, interpersonali
4. **Massimo 3 caratteristiche** architetturali prioritarie
5. **Modularità** (alta coesione, basso coupling) per manutenibilità
6. **Ogni scelta ha trade-off** - non esistono soluzioni perfette
7. **Misurazione continua** essenziale per qualità
8. **Sicurezza** progettata dall'inizio, non aggiunta dopo

Approccio Mentale:

- Pensa sempre ai **trade-off**
- Considera il **contesto** specifico
- Architettura è **arte di compromessi**
- Collega **esempi reali**: Netflix (microservizi), banche (sicurezza), Google (scalabilità)

Focus per Applicazioni Pratiche:

- **Quando** scegliere diversi pattern architetturali
- **Trade-off** cloud vs on-premise vs ibrido
- **Metriche qualità** software
- **Applicare** scalabilità, disponibilità, resilienza
- **Bilanciare** caratteristiche architetturali
- **Identificare** componenti con granularità appropriata