POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria Matematica

Tesi di Laurea Magistrale

Transizione dai Mainframe al Cloud: Modelli, Architetture Dati e Integrazione su Azure

Relatore
Prof. Daniele Apiletti

Correlatore
Simone Maiolo

Candidato
Sara Saffioti

Anno Accademico 2024-2025

A me stessa,
per aver dimostrato di
potercela fare davvero.

Sommario

La tesi affronta il tema della migrazione dei sistemi legacy da mainframe al cloud, analizzando strategie e implicazioni legate alla modernizzazione delle architetture dati. Viene presentata una panoramica dei principali approcci di migrazione (lift & shift, replatforming, refactoring, rebuild) e dei criteri che spingono verso l'adozione del cloud per garantire scalabilità, flessibilità e riduzione dei costi.

Un'attenzione particolare è rivolta alla piattaforma Microsoft Azure, illustrando come i suoi servizi possano supportare la transizione: dal data warehouse (PostgreSQL, Synapse Analytics), al data lake (Azure Data Lake Storage), fino all'architettura lakehouse (Databricks).

La parte applicativa si concentra su un progetto reale, in cui le pipeline implementate con Azure Data Factory orchestrano i flussi di dati secondo un approccio ELT (Extract–Load–Transform), includendo processi di pulizia, normalizzazione e auditing, accompagnati da meccanismi di archiviazione su Blob Storage e Data Lake.

I risultati evidenziano come la migrazione al cloud consenta di ridurre i costi, migliorare la governance e realizzare un sistema moderno e scalabile, in grado di supportare sia analisi operative sia analisi avanzate su dati storici. La tesi si conclude delineando possibili prospettive di evoluzione futura, che riguardano sia gli strumenti di gestione dei dati sia più in generale le architetture cloud.

Indice

Elenco delle figure

Introduzione

Questa tesi nasce dalla mia partecipazione a un progetto aziendale che ha previsto la migrazione di un sistema mainframe storico verso una piattaforma cloud moderna. Nel titolo è stato scelto il termine *transizione* invece che *migrazione*, perché il lavoro non si limita a descrivere il processo tecnico affrontato nel progetto, ma prende in considerazione anche il percorso storico ed evolutivo che ha portato, nel tempo, dal paradigma mainframe a quello cloud.

I mainframe hanno rappresentato per decenni un pilastro dei sistemi informativi e continuano ancora oggi a essere utilizzati in diversi contesti. La loro centralità è legata alla capacità di garantire affidabilità, sicurezza e potenza di calcolo, aspetti fondamentali soprattutto in settori come banche, sanità, assicurazioni e pubblica amministrazione. Negli ultimi anni, però, sono emersi limiti sempre più evidenti: costi di manutenzione elevati, rigidità architetturale e difficoltà di integrazione con tecnologie moderne. Per questo motivo, molte organizzazioni hanno intrapreso percorsi di modernizzazione o migrazione, spesso in maniera graduale, fino a un progressivo spostamento verso il cloud.

Il paradigma del cloud computing rappresenta infatti un modello alternativo, capace di offrire flessibilità e scalabilità maggiori, riducendo al tempo stesso gli investimenti iniziali e permettendo una gestione dei costi più trasparente attraverso il modello pay-per-use. Nel corso della tesi vedremo come il cloud renda possibile tempi di implementazione più rapidi, una maggiore reattività ai picchi di domanda e soluzioni integrate per la resilienza e la continuità operativa. Non mancano tuttavia sfide importanti: dalla sicurezza e conformità normativa, al rischio di dipendenza da un singolo fornitore, fino alla complessità di garantire pienamente le proprietà ACID nelle applicazioni transazionali. La protezione dei dati rimane quindi un aspetto cruciale, affrontato con modelli come la Zero Trust Security e con l'uso crescente dell'intelligenza artificiale per individuare in modo proattivo attacchi informatici.

In questo contesto, Microsoft Azure si posiziona come una delle piattaforme cloud più complete e diffuse. Il progetto descritto in questa tesi si basa proprio su Azure: il portale su cui stiamo trasferendo le funzionalità del mainframe è infatti costruito su questa

piattaforma. Approfondiremo i servizi che offre e i motivi che hanno portato a sceglierla rispetto ad alternative come Google Cloud e Amazon Web Services. Tra i suoi principali punti di forza troviamo l'integrazione con l'ecosistema Microsoft, già molto diffuso nelle imprese, la rete globale di data center, il supporto a soluzioni ibride e le certificazioni di sicurezza e compliance, che la rendono particolarmente adatta a scenari enterprise e a percorsi graduali di transizione dal legacy al cloud.

Il percorso di migrazione non è mai un processo unico e lineare, ma può seguire strategie diverse. Vedremo come la letteratura e gli analisti, tra cui Gartner, identifichino diversi approcci, dal semplice rehosting alla ricostruzione completa dei sistemi, e come ciascuno comporti compromessi specifici in termini di tempi, costi e benefici. Nel progetto aziendale a cui ho preso parte, la strategia adottata ha seguito un approccio vicino alla ricostruzione, con architetture moderne e un rilascio graduale, pensato per bilanciare tempi e rischi della transizione.

Un aspetto centrale del progetto riguarda l'uso di Azure Data Factory (ADF). Questo strumento ha reso possibile l'automazione dei flussi informativi, orchestrando in maniera scalabile i processi di ingestione, trasformazione e caricamento dei dati. Grazie ad ADF è stato possibile standardizzare procedure complesse, ridurre il ricorso a codice personalizzato e garantire la qualità dei dati attraverso controlli, validazioni e tracciamenti. Nel corso della tesi vedremo come le pipeline implementate si integrino nell'architettura complessiva, diventando un elemento chiave della transizione dal mainframe a un sistema più moderno e flessibile.

Un ulteriore tema di rilievo riguarda le architetture dati, che costituiscono la base di ogni ecosistema informativo. Partiremo dal data warehouse, che ha a lungo rappresentato lo standard per fornire dati strutturati e affidabili a supporto dei processi decisionali. Analizzeremo poi il paradigma del data lake, nato per raccogliere e gestire dati eterogenei in grandi volumi e in forma grezza. Infine, approfondiremo il concetto di lakehouse, che unisce i vantaggi dei due approcci, consentendo di gestire in un unico ambiente dati strutturati e non strutturati, riducendo le duplicazioni e semplificando l'interoperabilità con strumenti analitici. Nel progetto, vedremo come questi modelli siano stati applicati attraverso le pipeline sviluppate in Azure Data Factory, con livelli di data warehouse per i dati più recenti e uno storage di tipo data lake per lo storico, e con prospettive di evoluzione verso soluzioni di lakehouse supportate da strumenti come Databricks e Delta Lake.

In sintesi, questa tesi intende offrire una visione della transizione dai mainframe al cloud, combinando aspetti teorici e pratici. L'obiettivo è mostrare come la migrazione non sia soltanto un cambiamento tecnologico, ma un processo più ampio che coinvolge architetture, dati e governance, permettendo alle imprese di cogliere appieno i benefici del cloud in termini di efficienza, flessibilità e innovazione

Capitolo 1

Il Mainframe: evoluzione, vantaggi e limiti

Il mainframe è un sistema informatico centralizzato, dotato di potenti unità di elaborazione, una grande capacità di memoria e avanzati dispositivi di input/output. Grazie a questa struttura può gestire milioni di transazioni ogni giorno, servire contemporaneamente un numero enorme di utenti ed elaborare grandi volumi di dati, garantendo al tempo stesso sicurezza e affidabilità molto elevate [1]. Un aspetto distintivo è la capacità di rimanere operativo anche in caso di malfunzionamenti, resa possibile dall'impiego di componenti hardware ridondanti [1]. Per queste ragioni i mainframe sono largamente utilizzati in settori critici come banche, assicurazioni, logistica e sanità, contesti nei quali stabilità e continuità del servizio risultano indispensabili.

Dal punto di vista fisico si tratta di sistemi di grandi dimensioni, di gran lunga maggiori rispetto a quelli dei server standard. Se questi ultimi possono essere paragonati a rack da pochi centimetri di altezza, i mainframe occupano interi armadi e richiedono infrastrutture dedicate per alimentazione e raffreddamento.

Di seguito vengono riportate alcune delle principali attività gestite da questi sistemi in diversi contesti operativi:

- Elaborazione batch: esecuzione automatica di grandi volumi di dati senza alcun intervento umano, come ad esempio il calcolo degli stipendi di migliaia di dipendenti, tipicamente programmato negli orari notturni;
- Elaborazione transazionale ad alto volume: gestione in tempo reale di operazioni come prelievi da sportelli ATM, pagamenti con carta o bonifici bancari;
- Analisi dei dati: elaborazione di dataset storici e in tempo reale per ottenere informazioni, ad esempio l'analisi dei comportamenti dei clienti;

- Comunicazioni ad alta larghezza di banda: gestione e sincronizzazione di flussi di dati tra sistemi distribuiti a livello globale, come tra filiali bancarie localizzate in diversi Paesi;

- Pianificazione delle risorse aziendali (ERP): integrazione di processi aziendali complessi come produzione, contabilità e gestione delle risorse umane all'interno di un'unica infrastruttura centralizzata.

Figura 1.1. Sala server moderna che rappresenta l'ambiente fisico in cui i mainframe operano [14]

Nonostante l'avvento del cloud computing ci abbiamo portato a pensare al mainframe come ad una tecnologia datata, in realtà questa architettura si è evoluta per adattarsi alle esigenze moderne. Infatti, i sistemi legacy di nuova generazione sono in grado di integrare strumenti moderni, come container, algoritmi di machine learning e interfacce basate su API REST. Di conseguenza, il mainframe continua a occupare un ruolo centrale in molti contesti aziendali [1].

Secondo IBM, la maggior parte delle principali banche mondiali, compagnie aeree e rivenditori globali si affidano ancora al mainframe come piattaforma principale per i propri workload mission-critical [1].

Tuttavia, il mainframe presenta alcuni limiti: costi elevati di manutenzione, scarsa scalabilità, tempi di rilascio lenti e difficoltà nel reperire personale specializzato in sistemi legacy. Per questi motivi molte aziende stanno decidendo di intraprendere una migrazione verso tecnologie più moderne, ossia verso il cloud. In casi come questi, si va a riprogettare processi e database aziendali in un nuovo ambiente, mantenendo la continuità dei servizi ma riducendo i vincoli del mainframe.

1.1 Storia del mainframe

I primi mainframe erano macchine enormi, che occupavano interi locali con un'area di centinaia di metri quadrati. Necessitavano di grandi quantità di energia elettrica, impianti di raffreddamento dedicati e numerosi dispositivi di input/output [1][2].

La storia evolutiva che ha interessato i sistemi mainframe è legata principalmente a IBM, anche se prima della sua entrata in scena furono sviluppati altri calcolatori che preparavano alla nascita di questa tecnologia.

Due possono essere considerati i precursori del mainframe:

- L'Harvard Mark, un calcolatore elettromeccanico, venne sviluppato nel 1944, durante la Seconda Guerra Mondiale, per eseguire complesse operazioni matematiche e militari. [1]
- L'ENIAC fu il primo calcolatore elettronico universale e fu sviluppato dall'Università della Pennsylvania nel 1946, all'inizio per eseguire calcoli matematici e successivamente solo per usi scientifici. [12][13]

Non si trattava di mainframe veri e propri, ma di calcolatori che però anticipavano alcune delle principali caratteristiche che appartengono ai sistema legacy, ovvero le grandi dimensioni, la centralizzazione delle elaborazioni e l'utilizzo simultaneo da parte di più utenti [2]. Il primissimo mainframe commerciale della storia viene introdotto nel 1951, con nome di UNIVAC I. Si trattava di un calcolatore interamente elettronico ad uso aziendale e governativo. Il suo primo esemplare venne installato presso l'U.S. Census Bureau per l'elaborazione dei dati demografici. [12][13]

Due anni dopo, IBM entrò in scena con l'IBM Model 701, la sua prima macchina dedicata all'elaborazione dei dati: era molto più veloce rispetto ai predecessori e disponeva di maggiore capacità di calcolo e memoria [1][2].

Il vero punto di svolta arrivò nel 1964 con la famiglia IBM System/360, che rese possibile eseguire lo stesso software su più modelli della stessa serie. Per la prima volta venne superata la frammentazione che caratterizzava i sistemi precedenti, portando alla nascita di un'architettura compatibile e scalabile [1].

La virtualizzazione è un'innovazione che oggi associamo al cloud ma in realtà nacque proprio grazie al mainframe: infatti IBM sviluppò un hypervisor in grado di suddividere la macchina in più ambienti virtuali indipendenti, ciascuno con il proprio sistema operativo. Questa soluzione aumentò l'efficienza e rese possibile il lavoro simultaneo di più utenti sullo stesso sistema [1][3].

Nel 1970 venne introdotto il System/370, che segnò il passaggio dalla memoria a nuclei magnetici ai chip di silicio. Le prestazioni migliorarono, lo spazio occupato si ridusse e i sistemi divennero sempre più potenti. In quegli anni anche aziende come Fujitsu, Hitachi e Siemens entrarono nel mercato, portando innovazioni nell'I/O, nella capacità di memoria e nell'impiego di processori multipli [2].

Negli anni '90 la diffusione dei personal computer e dei server spinse molti analisti a sostenere che il mainframe fosse ormai destinato a scomparire. *"L'ultimo mainframe sarà spento il 15 marzo 1996"* è la frase che venne attribuita a Stewart Alsop II, allora caporedattore della rivista InfoWorld [2]. In realtà, questa previsione si rivelò sbagliata, infatti questi sistemi continuarono a evolversi, confermandosi insostituibili in settori in cui affidabilità, sicurezza e scalabilità sono requisiti essenziali [2].

Il fatto che è i mainframe non si siano estinti è confermato dalla nascita dell'IBM z16 nel 2022. Si tratta di un mainframe moderno, dotato del processore Telum, con acceleratori di intelligenza artificiale in grado di eseguire analisi predittive e rilevare comportamenti anomali in tempo reale, gestendo milioni di transazioni simultanee anche in contesti ibridi o cloud-native [1][2].

1.2 Caratteristiche del mainframe moderno

I primi mainframe disponevano di un'unica CPU centrale: tutte le operazioni venivano gestite da unico punto di calcolo, comportando così evidenti limiti in termini di scalabilità. Invece, i mainframe moderni hanno invece un'architettura più vantaggiosa: sono dotati di CPC (Central Processor Complex), ossia un insieme modulare di processori e sottosistemi, ognuno dei quali progettato per svolgere un compito specifico. In questo modo è possibile distribuire i carichi di lavoro e garantire continuità operativa anche in scenari complessi [1][2].

In particolare, il CP (Central Processor) può essere considerato il vero e proprio "cervello" del mainframe moderno: gestisce le operazioni generali e tutte le elaborazioni di base. Accanto al CP esistono anche processori specializzati, progettati per svolgere compiti più mirati e alleggerire il CP, migliorando così l'efficienza complessiva del sistema:

- SAP (System Assist Processor): gestisce le operazioni di input/output e lo spostamento dei dati tra i vari sistemi [1];
- IFL (Integrated Facility for Linux): consente di eseguire carichi di lavoro Linux in modo isolato e ottimizzato sul mainframe, permettendo di consolidare molte VM

Linux su un unico sistema [1];

- zIIP (z Integrated Information Processor): esegue carichi di lavoro specifici come DB2, XML, analisi, container o API REST. Con gli ultimi aggiornamenti, se non è presente uno zAAP, i carichi Java e XML possono essere eseguiti su zIIP [2][3];
- zAAP (z Application Assist Processor): dedicato ai carichi Java e XML; con l'evoluzione dei sistemi molte delle sue funzioni sono state integrate in zIIP [2][3].

Questo approccio non solo migliora le prestazioni e la scalabilità, ma consente anche di risparmiare sui costi delle licenze software. Infatti, nelle architetture IBM il costo delle licenze è calcolato principalmente in base al numero di CP utilizzati e non sui processori specializzati. Di conseguenza, se il carico di lavoro aumenta, invece di aggiungere nuovi CP (che farebbero crescere il costo delle licenze) è possibile spostare parte del carico sui processori specializzati, mantenendo i costi sotto controllo [1][2].

Accanto ai processori, un mainframe è dotato di componenti dedicati alla sicurezza e allo storage:

- Schede di rete: collegano i mainframe con altri sistemi in modo sicuro e veloce, proteggendo i dati da accessi non autorizzati e da eventuali rallentamenti durante il trasferimento [1][4];
- Moduli crittografici hardware: sono dotati di algoritmi avanzati, che cifrano i dati in modo più rapido rispetto ad una normale CPU [1][4][5];
- Unità di compressione: ottimizzano lo spazio in cui vengono archiviati i dati, aumentando così la velocità con cui è possibile leggerli e trasferirli [1].

Grazie a questa architettura, i mainframe sono in grado di gestire milioni di transazioni simultanee, evitando interruzioni e offrendo livelli di affidabilità e sicurezza che altri sistemi difficilmente possono garantire.

1.3 Integrazione con le tecnologie moderne

Anche se molto spesso vengono considerate macchine obsolete, i mainframe hanno continuato ad evolversi nel tempo integrandosi con le tecnologie più moderne.

- Possono comunicare con le piattaforme cloud, permettendo di spostare dati e processi tra i due ambienti e creando così un sistema ibrido che unisce l'affidabilità tipica del mainframe con la flessibilità del cloud [4];
- Possono supportare attività di business intelligence, infatti sono capaci di analizzare enormi quantità di dati, sia quelli storici che quelli prodotti in tempo reale, per prendere decisioni più rapide e consapevoli [2];

- Alcuni modelli mainframe includono già funzioni di intelligenza artificiale: ad esempio l'IBM z16 con il suo processore Telum, che riesce a riconoscere tentativi di frode o comportamenti anomali in pochi millisecondi [5];

- Guardano anche al futuro del calcolo quantistico, infatti utilizzano sistemi di crittografia "quantum-safe", progettati per proteggere le informazioni da possibili minacce che i computer quantistici potrebbero introdurre [4].

Questa continua capacità di adattamento dimostra perché i mainframe non sono scomparsi, ma rimangono ancora oggi una colonna portante delle infrastrutture IT di tante aziende.

1.4 Vantaggi del mainframe

I principali vantaggi che caratterizzano i mainframe sono:

- Affidabilità: i mainframe sono concepiti per garantire un funzionamento continuo e presentano tassi di guasto nettamente inferiori rispetto ad altri sistemi informatici. Questo risultato si ottiene grazie a meccanismi di controllo interni che monitorano costantemente hardware e software: in presenza di anomalie, il problema può essere corretto immediatamente o isolato, evitando che si propaghi nel resto dell'infrastruttura. La capacità di autocorrezione rende questi sistemi particolarmente adatti ad applicazioni critiche, dove anche brevi interruzioni potrebbero avere effetti rilevanti [1].

- Disponibilità: tale caratteristica è resa possibile dalla ridondanza di componenti come alimentatori, schede di rete, processori e dischi, che sono duplicati o addirittura triplicati. In caso di guasto, la copia entra subito in funzione senza causare disservizi agli utenti. A ciò si aggiungono i sistemi di disaster recovery, che replicano i dati in data center remoti e consentono la riallocazione dei carichi di lavoro anche in scenari estremi come incendi o interruzioni di corrente [2].

- Gestione e servizio: la presenza di strumenti diagnostici avanzati facilita l'amministrazione dei mainframe, permettendo di individuare rapidamente l'origine di eventuali malfunzionamenti. Anche in presenza di problemi, il sistema resta attivo e i tecnici possono sostituire i componenti difettosi senza interrompere le attività in corso. Inoltre, molte operazioni di manutenzione sono automatizzate, riducendo la necessità di intervento manuale e offrendo agli amministratori un supporto immediato [3].

- Prestazioni e ottimizzazione: le architetture moderne consentono al mainframe di eseguire più sistemi operativi in parallelo, come z/OS e Linux on Z, e di gestire

carichi di lavoro molto diversi tra loro. Alcuni processori hanno funzioni specializzate, ad esempio crittografia, compressione dei dati o gestione delle operazioni di input/output, che alleggeriscono la CPU principale e mantengono alte le prestazioni anche con milioni di transazioni simultanee [1][4].

- Scalabilità: la possibilità di aumentare le risorse senza fermare il sistema è un tratto fondamentale del mainframe. Questi sistemi supportano sia la scalabilità verticale (aggiunta di risorse in una singola macchina) sia quella orizzontale (distribuzione dei carichi su più macchine). È possibile quindi integrare nuovi processori, memoria o capacità di storage a caldo, mantenendo elevato il throughput e dei tempi di risposta ridotti [2].

- Sicurezza: la protezione dei dati è centrale nella progettazione dei mainframe. Essi includono moduli hardware dedicati alla crittografia e software che monitorano continuamente gli accessi. Tecniche basate su intelligenza artificiale e machine learning consentono di rilevare comportamenti anomali e di attivare misure difensive in tempo reale. Grazie a tali funzioni, i mainframe possono gestire miliardi di transazioni online in modo sicuro e conforme alle normative [1][5].

- Resilienza: questi sistemi raggiungono la continuità operativa grazie ad una progettazione che assicura il funzionamento anche in condizioni estreme. Ogni componente essenziale, come CPU, sistemi di raffreddamento, alimentatori e canali di I/O, è duplicato e progettato per operare in modo indipendente. Alcuni modelli sono inoltre certificati per resistere a forti variazioni ambientali di temperatura e umidità, caratteristica che li rende particolarmente adatti a settori sensibili come banche, assicurazioni, trasporti e pubblica amministrazione [3][4].

1.5 Principali ambiti di utilizzo dei mainframe

Nel settore industriale e manifatturiero i mainframe sono impiegati per attività complesse: controllo delle linee produttive, analisi dei dati provenienti da sensori e dispositivi IoT e manutenzione predittiva degli impianti. In questo campo si rivelano particolarmente utili perché consentono di monitorare processi produttivi critici in tempo reale, riducendo al minimo i rischi di fermo macchina [4][5].

Strutture sanitarie pubbliche e private (ospedali, ASL e compagnie assicurative) gestiscono ogni giorno grandi quantità di dati clinici, amministrativi e anagrafici. I mainframe vengono scelti perché permettono di aggiornare in tempo reale le cartelle cliniche e le terapie disponibili. Garantiscono il rispetto di normative stringenti sulla privacy, come GDPR in Europa o HIPAA negli Stati Uniti, e assicurano continuità operativa anche in situazioni di emergenza sanitaria o in caso di picchi di accesso.

Le banche e le società che operano nei servizi finanziari sono tra i principali utilizzatori dei mainframe. Ogni giorno devono gestire milioni di operazioni: bonifici, prelievi da sportelli ATM, pagamenti elettronici, investimenti e calcoli sugli interessi. In questi scenari il mainframe rimane essenziale poichè assicura continuità operativa 24 ore su 24, protegge i dati sensibili da attacchi informatici, elabora in tempo reale enormi volumi di transazioni, ed esegue attività di tracciabilità, utili per i controlli interni e la conformità a regolamenti come PCI-DSS o Basel III [1][2].

Anche le compagnie assicurative si affidano ai mainframe per gestire polizze, richieste di risarcimento, rimborsi e analisi del rischio, tutti ambiti in cui stabilità e continuità non possono mancare.

Compagnie ferroviarie, aeree e navali utilizzano i mainframe per la gestione di prenotazioni, programmazione orari, monitoraggio flotte e tracciamento bagagli. Questi sistemi permettono di aggiornare rapidamente informazioni come ritardi, variazioni di gate o disponibilità dei posti, mantenere affidabili i sistemi di controllo e interconnettersi con partner esterni, tra cui aeroporti, compagnie collegate e dogane [2].

Le amministrazioni pubbliche fanno ampio uso dei mainframe per erogare servizi critici destinati a milioni di cittadini. Alcuni esempi sono la gestione di tasse e rimborsi fiscali da parte delle Agenzie delle Entrate, l'erogazione di pensioni e sussidi da parte degli enti previdenziali (come INPS o IRS), il trattamento di fascicoli giudiziari, denunce e dati biometrici da parte delle forze dell'ordine e i servizi anagrafici come rilascio di passaporti, certificati e documenti d'identità. In questi contesti, il mainframe è apprezzato per la capacità di resistere a tentativi di attacco informatico, proteggere dati sensibili e gestire milioni di richieste contemporanee [2][4].

Le grandi catene commerciali e i marketplace online richiedono sistemi capaci di gestire ordini, resi, logistica e scorte in tempo reale. Nei periodi di picco, come il Black Friday o le festività natalizie, i mainframe garantiscono stabilità del servizio, anche con milioni di clienti collegati nello stesso momento, protezione delle transazioni e dei dati di pagamento, aggiornamento immediato degli inventari e delle offerte promozionali e integrazione fluida con applicazioni cloud e piattaforme web [1][3].

1.6 Limiti e problemi attuali del mainframe

Abbiamo dedotto che i mainframe sono considerati sistemi fondamentali in diversi settori, ma è importante sottolineare che presentano anche una serie di criticità. Alcune derivano

dai costi e dalla complessità tecnica, altre dal confronto con le tecnologie moderne, ad esempio il cloud, che offre maggiore flessibilità e scalabilità [1][6].

- Complessità gestionale: questi sistemi devono essere gestiti da personale con competenze specifiche su ambienti basati su linguaggi datati, ma è davvero difficile trovare questo tipo di figure. Attività come configurazione, monitoraggio delle performance, sicurezza e troubleshooting non possono essere eseguite da chiunque abbia esperienza nel settore informatico, ma necessitano di personale specializzato nei mainframe, e la scarsità di tali competenze può comportare dei rischi operativi [3][6]. Inoltre, la dipendenza da pochi esperti può rallentare eventuali processi di aggiornamento o modernizzazione del mainframe, rendendo l'organizzazione più vulnerabile in caso di turnover.

- Dipendenza dal software legacy :nelle aziende le applicazioni fondamentali sviluppate all'interno dei loro sistemi mainframe si basano su linguaggi storici come COBOL o PL/I. Si tratta di programmi che gestiscono processi cruciali all'interno dell'organizzazione, ma che risultano difficili da aggiornare o integrare con sistemi moderni, poiché all'interno di ogni programma mainframe sono state effettuate modifiche in tempi diversi o da team diversi, che non sono state documentate. La mancanza di programmatori esperti in questi tipi di linguaggi rappresenta un ulteriore problema: in caso di malfunzionamenti o necessità di modifiche, le imprese non sanno bene come comportarsi [2][6].

- Rigidità nell'adozione di nuove tecnologie: nonostante negli anni i mainframe siano diventati più integrabili con il cloud e con API moderne, la loro architettura monolitica continua a rappresentare un limite. L'introduzione di soluzioni cloud-native, come microservizi e container, risulta spesso complessa da mettere in atto, rallentando così l'adozione di tecnologie innovative [4][6].

- Scalabilità limitata: tradizionalmente i mainframe sono stati progettati per crescere in verticale, ossia aumentando le risorse (CPU, memoria, storage) all'interno di una singola macchina. Questo modello, sebbene potente, presenta limiti tecnici ed economici: oltre una certa soglia i costi crescono rapidamente e l'ampliamento richiede tempi lunghi e procedure complesse. Al contrario, le architetture cloud moderne sfruttano la scalabilità orizzontale, distribuendo i carichi di lavoro su molte macchine meno costose e più facilmente sostituibili [6].

- Costo elevato:l'acquisto dell'hardware è solo l'inizio: l'installazione e la configurazione sono attività complesse, e quindi richiedono personale esperto; a ciò si aggiungono i costi di licenze, manutenzione, aggiornamenti software e gestione ordinaria. Anche i consumi energetici e i sistemi di raffreddamento prevedono costi elevati, rendendo i mainframe accessibili quasi esclusivamente a grandi organizzazioni o enti pubblici [2][6].

- Consumi energetici e sostenibilità ambientale: un mainframe richiede molta energia non solo per funzionare, ma anche per alimentare i sistemi di raffreddamento. Di conseguenza aumentano sia le spese operative sia l'impatto ambientale, soprattutto se confrontati con data center cloud moderni, progettati per ridurre i consumi [5][6].

- Dipendenza dal fornitore: un altro elemento di criticità è il cosiddetto "vendor lock-in". Chi adotta un mainframe dipende dal produttore, quasi sempre IBM, per hardware, aggiornamenti, software e servizi di supporto. Questo riduce la flessibilità e può determinare costi aggiuntivi in caso di cambi di contratto o di strategia aziendale [4].

- Sfide nella modernizzazione: la migrazione da mainframe a soluzioni ibride o completamente cloud comporta processi lunghi e complessi. Spesso è necessaria la riscrittura del codice legacy e la gestione di problemi legati a sicurezza, interoperabilità e trasferimento dei dati. Ciò comporta costi elevati e il rischio di interruzioni dei servizi critici, motivo per cui molte aziende preferiscono mantenere i mainframe e puntare su un'integrazione graduale [1][6].

1.7 Mercato attuale e prospettive future

Molti considerano i mainframe una tecnologia destinata a scomparire, ma le analisi di mercato indicano l'opposto. Diversi studi hanno mostrato che l'impiego dei mainframe è in continua crescita: Straits Research ha stimato un valore di circa 3 miliardi di dollari nel 2024, con la possibilità di superare i 4,5 miliardi entro il 2032 (crescita media annua circa 7%) [7]. Mordor Intelligence ha fatto delle previsioni simili, ipotizzando che entro il 2033 il settore possa arrivare attorno ai 6,6 miliardi di dollari, con un'espansione annua vicina all'8% [8].

Accanto al mercato dei sistemi, cresce anche quello dei servizi di aggiornamento. Secondo MarketsandMarkets, l'integrazione e la modernizzazione dei mainframe passeranno da circa 8 miliardi nel 2025 a oltre 13 miliardi entro il 2030, con una crescita annua superiore al 9% [9]. Ciò significa che le imprese non abbandonano i mainframe, ma li rendono più flessibili e compatibili con ambienti moderni. Sul fronte tecnologico, i produttori continuano a innovare: IBM, ad esempio, ha introdotto nel 2022 il modello z16, con acceleratori di intelligenza artificiale integrati, invece per il 2025 è previsto lo z17, progettato per contesti cloud ibridi e dotato di crittografia "quantum-safe" per resistere alle minacce informatiche future [10].

Questi elementi mostrano come i mainframe non siano affatto destinato ad estinguersi: la loro evoluzione li rende ancora oggi fondamentali in settori che non possono rinunciare ad

affidabilità e sicurezza, come banche, sanità e pubblica amministrazione.

Figura 1.2. IBM z17, progettato per il cloud ibrido e dotato di crittografia quantum-safe [16]

1.8 Modernizzazione graduale e ruolo dell'IA

Oggi il tema centrale non è più l'abbandono del mainframe, ma la sua evoluzione graduale. Le imprese, infatti, tendono a mantenere questi sistemi affiancandoli a tecnologie moderne, così da contenere i costi e ridurre i rischi legati a una sostituzione completa. Tra le soluzioni più diffuse vi sono l'uso di API per mettere in comunicazione le applicazioni legacy con piattaforme cloud e l'impiego di container, che permettono di isolare i carichi di lavoro e renderli più flessibili.

In questo percorso, anche l'intelligenza artificiale generativa ha un ruolo sempre più rilevante. Alcuni strumenti, ad esempio, sono in grado di analizzare applicazioni sviluppate in linguaggi datati come COBOL, producendo automaticamente documentazione e rendendo più comprensibile un codice che spesso non ha più riferimenti aggiornati [11]. Grazie a questo approccio, diventa più semplice integrare il mainframe con pratiche DevOps e, allo stesso tempo, continuare a beneficiare dell'affidabilità che da sempre lo caratterizza.

1.9 Persistenza dei limiti e scenario progettuale

Nonostante le innovazioni introdotte negli ultimi anni, i mainframe continuano a presentare limiti strutturali che non possono essere ignorati. I costi di manutenzione rimangono molto elevati, la gestione richiede competenze difficili da trovare e l'integrazione con architetture cloud-native resta molto complessa. Questi aspetti portano molte imprese di scegliere di pianificare una dismissione graduale dei sistemi legacy.

Un esempio concreto è rappresentato dalla piattaforma di gestione dei pezzi di ricambio ancora in uso nella nostra azienda. Questo sistema funge da archivio centrale per i dati anagrafici dei pezzi, le sostituzioni, i kit, le note tecniche e i costi standard associati. I dati vengono raccolti da altri sistemi aziendali (come quelli di progettazione e approvvigionamento) e messi a disposizione di varie applicazioni, a supporto di attività operative come gestione dei magazzini, flussi finanziari e garanzie. Un aspetto rilevante è l'interdipendenza tra i dati tecnici e quelli economici: ogni modifica di prezzo o introduzione di un nuovo componente si riflette direttamente sulla possibilità di ordinarlo e commercializzarlo. Negli anni, alcune funzionalità, come ordini, fatturazione e magazzino, sono già state migrate su soluzioni più moderne. Nonostante questo, il sistema rimane legato al mainframe, con conseguenze significative in termini di rigidità, costi e complessità di gestione.

Queste criticità emergono chiaramente nel nostro progetto, che ha analizzato la situazione attuale e la visione futura del sistema. Attualmente il sistema opera su mainframe, con logiche separate per la gestione dei processi legati ai pezzi e ai costi. Questa frammentazione non solo aumenta la complessità, ma accresce anche i costi operativi, già in crescita a causa delle spese per l'hardware di manutenzione del mainframe. Inoltre, l'integrazione con più sistemi esterni per lo scambio dei dati rende l'infrastruttura meno agile e più difficile da mantenere.

La soluzione futura punta invece a trasferire queste funzionalità su piattaforme più moderne, migliorando la gestione dei flussi di interfaccia e rivedendo le procedure attualmente in uso. Il risultato atteso è una riduzione dei costi grazie a un'infrastruttura più snella, minori attività di manutenzione su interfacce complesse e una maggiore flessibilità nell'evoluzione dei processi aziendali.

In questo modo, il progetto conferma che la dismissione del mainframe non è soltanto una scelta tecnologica, ma una vera necessità per garantire sostenibilità economica, semplificazione delle logiche gestionali e capacità di adattamento alle esigenze future.

Capitolo 2

Cloud Computing: fondamenti, modelli e servizi

Internet ha cambiato radicalmente il mondo dell'informatica: stiamo passando da un'epoca in cui tutti gli utenti possedevano un computer ad una in cui gli utenti non lo possiedono più ma possono avere accesso all'hardware e al software da remoto.

In questo contesto si inserisce il cloud computing, che rende disponibili risorse e servizi, come sistemi operativi, applicazioni, archiviazione e capacità di elaborazione, a chiunque disponga di una connessione a Internet, da qualsiasi luogo e tramite qualsiasi dispositivo. I servizi vengono attivati solo quando richiesti (on demand) e il pagamento avviene in base all'effettivo consumo, con un modello simile a quello delle utenze domestiche.

Il vero vantaggio del cloud è che l'utente non deve acquistare, installare, gestire né provvedere alla manutenzione dell'infrastruttura hardware: tutto questo è interamente a carico del provider, che si occupa dei server ospitati in grandi data center distribuiti a livello globale e interconnessi tra loro. All'utente basta un dispositivo connesso a Internet per accedere ai programmi e ai dati memorizzati in remoto.

In questo modo gli utenti, o più in generale intere organizzazioni, possono concentrarsi sul proprio core business, attivando e gestendo le risorse in autonomia (self-service) e delegando attività aziendali periferiche al fornitore dei servizi cloud.

Ogni giorno utilizziamo servizi cloud senza rendercene conto, ad esempio quando inviamo un' e-mail tramite Gmail stiamo utilizzando un software di posta elettronica fornito dalla piattaforma cloud di Google.

Molteplici sono le aziende che forniscono questo servizio e che dispongono di milioni di

data center sparsi in tutto il mondo: le più rilevanti sono Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), IBM Cloud, Oracle Cloud [17].

Figura 2.1. Schema concettuale del cloud computing: le risorse presenti nei data center vengono distribuite tramite la nuvola e rese disponibili a diversi dispositivi e applicazioni (sanità, business, mobile, IoT, automotive, realtà virtuale, gaming) [40]

2.1 Storia del cloud computing

Il termine "cloud computing" è stato utilizzato per la prima volta probabilmente da Eric Schmidt nel suo discorso sulle conferenze sulle strategie per i motori di ricerca nel 2006 [19].

Negli anni a seguire sono stati molti i ricercatori che hanno dato una definizione di cloud computing. La più completa è senza dubbio quella fornita dal National Institute of Standards and Technology (NIST) degli Stati Uniti, poiché oltre a descrivere il concetto di cloud computing, sottolinea le sue caratteristiche fondamentali:

"Il cloud computing è un modello per consentire un comodo accesso alla rete on-demand a un pool condiviso di risorse informatiche personalizzabili (ad esempio, reti, server, storage, applicazioni e servizi) che possono essere rapidamente fornite e rilasciate con uno sforzo di gestione minimo o l'interazione con il fornitore di servizi". [18]

Dopo queste prime definizioni teoriche il cloud computing ha iniziato a concretizzarsi anche a livello commerciale. In particolare, il primo vero servizio di cloud computing moderno, per come lo intendiamo oggi, è stato lanciato da Amazon tramite Amazon Web Services (AWS) nel 2006, introducendo Amazon S3 (Simple Storage Service) e Amazon EC2 (Elastic Compute Cloud) [2].

Nonostante il cloud computing si sia affermato come modello tecnologico vero e proprio

solo a partire dal 2006, il suo concetto può essere ricondotto già agli anni '60, quando, seppur con caratteristiche differenti, si parlava di Utility Computing, Grid Computing e Autonomic Computing [18].

Nei paragrafi successivi verranno analizzati questi paradigmi, per poi descrivere come la loro evoluzione abbia condotto all'attuale modello di cloud computing.

2.1.1 Dal mainframe all' Utility Computing

Negli anni '50 un'università degli Stati Uniti (MIT) installò uno dei primi computer mainframe, ossia l'IBM 704, per scopi di ricerca scientifica. I processi di calcolo eseguiti da questo computer erano molto lenti: il programma da eseguire veniva inizialmente scritto su delle schede perforate, che poi venivano consegnate al centro di calcolo; a questo punto il programma veniva messo in coda insieme ad altri, creati da altri utenti. Il mainframe poteva eseguire un programma alla volta e ciò portava a lunghi tempi di attesa. Come se non bastasse, in caso di fallimento del programma, causato da errori di sintassi, bisognava rifare tutto il procedimento da capo [21].

Allora l'informatico John McCarthy ebbe l'idea di collegare, tramite più terminali, il computer centrale agli uffici dei docenti; in questo modo più utenti, ciascuno dal proprio terminale, potevano eseguire il proprio programma. Anche qui il mainframe eseguiva un programma alla volta, ma per un tempo lungo frazioni di secondi, dando così l'impressione agli utenti di operare simultaneamente (time-sharing). Questa proposta venne realizzata tramite il Compatible Time Sharing System (CTSS), ossia il primo sistema operativo di time-sharing funzionante [21].

Poco più tardi, durante le celebrazioni del centenario del MIT, lo stesso McCarthy introdusse il concetto di "Utility Computing", proponendo di trasformare la condivisione dei computer in un servizio pagabile a consumo. L'idea era quella di "affittare" le risorse da un fornitore, in qualsiasi momento e in qualsiasi luogo, pagando a consumo senza dover acquistare l'infrastruttura, come succedeva, ad esempio, nei servizi che forniscono acqua, elettricità, telefonia. A fine anni '90 HP ha lanciato l'Utility data Center, per fornire servizi di fatturazione IP, e PolyServe Inc. ha sviluppato un sistema clusterizzato, ottimizzato per storage di massa e calcolo ad alte prestazioni [18].

L'Utility Computing ha anticipato il modello del cloud, introducendo l'idea del pagamento a consumo, ma senza offrire la flessibilità, la scalabilità e la distribuzione globale che caratterizzano il cloud computing odierno.

2.1.2 Grid Computing

Negli anni '90 si afferma un nuovo un modello di calcolo, ossia il Grid Computing, in cui i vari nodi, che offrono potenza di calcolo, vengono collocati in diverse aree geografiche distanti tra loro. Ogni nodo appartiene ad un'organizzazione diversa che può stabilire delle proprie politiche di sicurezza, secondo le quali le proprie risorse possono essere accessibili. In questo modo eseguire dei compiti diventa più complesso, poiché un'organizzazione, per poter accedere ad altre risorse, deve adattarsi alle regole dell'organizzazione in cui le richiede.

Nel cloud computing, invece, tutte le risorse presenti nei vari nodi non sono indipendenti e vengono unificate in un'unica piattaforma, e ciò permette agli utenti un accesso trasparente .

Inoltre, il grid computing serve pochi clienti che però eseguono lavori complessi divisi tra più nodi, mentre il cloud computing serve migliaia o milioni di utenti contemporaneamente, gestendo piccoli compiti. [25]

2.1.3 Autonomic Computing

L' Autonomic Computing è un paradigma di calcolo proposto da IBM nel 2001, in cui il sistema si autogestisce, ovvero è in grado di effettuare compiti che normalmente verrebbero eseguiti da un professionista IT. La necessità di creare questo genere di dispositivi è dovuta al numero crescente di sistemi che richiedono una gestione manuale continua [18].

IBM ha progettato un sistema autonomo che può svolgere i seguenti compiti senza alcun intervento umano [18]:

- Self-Configuration (auto-configurazione): il sistema configura automaticamente i componenti;
- Self-Healing (auto-riparazione): ripara i problemi automaticamente;
- Self-Optimization (auto-ottimizzazione): ottimizza le prestazioni;
- Self-Protection (auto-protezione): protegge da attacchi o anomalie.

2.1.4 Dalla nascita all'evoluzione del cloud

Dopo aver analizzato i modelli che hanno anticipato il cloud computing, è utile riassumerne i contributi principali:

- L'Utility Computing aveva introdotto il concetto di risorse "a consumo";

- Il Grid Computing aveva mostrato come mettere insieme potenza di calcolo distribuita;
- L'Autonomic Computing ha aggiunto l'idea di sistemi auto-gestiti.

Il passo successivo è stato integrare tutti questi aspetti in un unico modello più maturo: il Cloud Computing, che dal 2006 in poi (con il lancio di Amazon AWS) si è imposto come paradigma dominante, unendo elasticità, scalabilità, automazione e accesso globale via Internet.

A partire dal 2007 il cloud computing ha iniziato a diffondersi grazie alla nascita di numerose piattaforme e servizi che hanno contribuito a trasformarlo da un semplice concetto teorico a modello tecnologico consolidato. Nei primi anni nascono le prime soluzioni open source, come ad esempio Eucalyptus e OpenNebula, che permisero di creare cloud privati (gestiti solo all'interno di una sola organizzazione) e cloud ibridi (che combinano risorse private e pubbliche offerte da diversi provider). Allo stesso tempo, alcune grandi aziende lanciano i propri progetti: nel 2010 nasce Microsoft Azure, che si affianca ad Amazon Web Services, già attivo dal 2006, e nel 2011 Apple introduce iCloud, portando il concetto di cloud fuori da contesto aziendale, mettendolo direttamente a disposizione degli utenti finali. Sempre nel 2011 viene fondata l'Open Networking Foundation (ONF), che sottolinea l'importanza delle reti programmabili per lo sviluppo di data center e infrastrutture cloud [34].

Negli anni successivi, l'offerta si amplia ulteriormente: altre grandi aziende come IBM, Oracle, Google e Salesforce entrano nel mercato del cloud, lanciando ognuno la propria piattaforma, rendendo così il mercato sempre più variegato e competitivo.

Dopo il 2015 si diffondono altre tecnologie che diventeranno centrali per il cloud moderno, come ad esempio i container e gli strumenti di orchestrazione, insieme ai primi modelli serverless, che segnano un'evoluzione rispetto all'idea di risorsa "a consumo" [35].

Tra il 2018 e il 2020, il cloud assume un ruolo sempre più strategico per le imprese: si inizia a parlare di multicloud e di cloud ibrido, che permettono di combinare servizi provenienti da provider diversi. Nello stesso periodo cresce anche l'attenzione verso aspetti come sicurezza, governance e gestione dei costi, che diventano priorità per le aziende che decidono di avviare processi di migrazione verso infrastrutture cloud.

Negli anni più recenti, dal 2021 al 2025, il cloud ha continuato ad evolversi, diventando un pilastro fondamentale nella trasformazione digitale. In questi anni si sono diffusi modelli come il multicloud e l'integrazione con l'intelligenza artificiale, che hanno reso il cloud sempre più strategico per imprese e istituzioni [36].

È quindi possibile affermare che dal 2007 al 2025 il cloud computing si sia evoluto in modo costante: dalle prime soluzioni frammentate, si è passati a un ecosistema globale, dominato da grandi provider e arricchito da un insieme di tecnologie e modelli che ne hanno fatto l'infrastruttura di riferimento per l'innovazione digitale.

2.2 Elementi del cloud computing

Tre sono gli elementi essenziali che caratterizzano il cloud computing:

- Data center: si tratta di edifici contenenti server che forniscono risorse di calcolo e di archiviazione, condivise da milioni di utenti, tramite una connessione a Internet;
- Connessione a Internet: la rete globale che collega gli utenti ai datacenter, permettendo l'accesso remoto ai servizi cloud da qualsiasi luogo;
- Terminali: dispositivi tramite i quali gli utenti si collegano ai servizi cloud.

Andando più nel dettaglio:

I data center che utilizzano un'architettura cloud sono sparsi in tutto il mondo ed ciuascuno di essi può ospitare da alcune centinaia fino a decine di migliaia di server. Dispongono di:

- sistemi di storage, con diverse configurazioni, ossia DAS (storage strettamente vicino al server per accedere più rapidamente ai dati utilizzati più di frequente), NAS (storage collegato a più server, che utilizza la rete principale per trasferire i dati) e SAN (archiviazione condivisa simile al NAS, ma che utilizza una rete separata dedicata solo ai dati);
- sistemi di alimentazione elettrica continua per evitare interruzioni o blackout;
- componenti ridondanti per garantire la continuità operativa dei server in caso di guasti;
- sistemi di raffreddamento per mantenere l'hardware ad una temperatura ottimale; in particolare vengono utilizzate tecnologie che utilizzano una combinazione di raffreddamento a liquido e ad aria [23].

Un altro fattore che ha sicuramente favorito lo sviluppo del cloud computing è stato l'aumento della larghezza di banda, accompagnato dalla riduzione dei costi legati all'utilizzo di essa. Ciò fa si che chiunque possa ricevere o inviare dati in tempo reale a partire da qualsiasi parte del mondo; in questo modo si può utilizzare qualsiasi tipo di servizio online, come se fosse installato sul proprio computer personale, anche se in realtà è ospitato in server lontani [21].

Infine, non vanno dimenticati i terminali, dispositivi che permettono di connetterci ai servizi e ai programmi che girano su un server remoto, ossia nel cloud. Non possiedono né una propria potenza di calcolo né programmi installati. Al contrario, un PC tradizionale è molto più complesso, poiché dispone di un sistema operativo, capacità di elaborazione interna e programmi installati localmente, di conseguenza può essere utilizzato anche in assenza di una connessione cloud. Sia i terminali che dispositivi come pc, tablet o smartphone possono accedere a servizi cloud, però i terminali si appoggiano completamente alla potenza di calcolo del cloud, lasciando a quest'ultimo il compito di eseguire tutte le applicazioni [21].

Oltre alla presenza di data center, Internet e terminali, il cloud computing si basa su protocolli di comunicazioni standard, ossia TCP/IP (Trasmission Control Protocol/ Internet Protocol), che costituiscono la base di Internet. In particolare, l'IP si occupa di indirizzare i pacchetti dei dati verso la destinazione corretta, mentre il TCP garantisce che questi pacchetti arrivino nell'ordine corretto e senza errori. Tutto ciò avviene indipendentemente dall'hardware e dal software utilizzati: dispositivi diversi possono comunicare tra di loro purchè supportino i protocolli TCP/IP [22].

2.3 Virtualizzazione nel cloud

Prima di analizzare il funzionamento di una piattaforma cloud, le sue proprietà, l'architettura e i servizi che essa mette a disposizione, è utile descrivere la tecnologia che ne costituisce le fondamenta: la virtualizzazione. Questa ha rivoluzionato il modo in cui le risorse informatiche vengono utilizzate e gestire.

Prima che venisse introdotta, un professionista IT doveva acquistare e configurare un server per ogni applicazione che voleva eseguire; ciò portava le aziende ad acquistare più server fisici che però venivano usati solo in parte. Con l'introduzione della virtualizzazione, invece, le risorse di un singolo server (processori, memoria, storage ecc.) vengano suddivise tra più ambienti virtuali. Ogni utente interagisce col proprio ambiente virtuale che percepisce come un sistema indipendente, mentre la complessità dell'hardware rimane nascosta [18][24].

In realtà, il concetto di virtualizzazione non è nato con il cloud moderno, ma affonda le sue radici già negli anni '60 e '70, quando IBM sviluppò i primi sistemi di time-sharing. Questi permettevano a più utenti di utilizzare simultaneamente lo stesso mainframe, ciascuno con il proprio ambiente isolato. In quegli anni si posero le basi teoriche di ciò che oggi conosciamo come macchine virtuali, le quali:

- simulano il funzionamento di più computer indipendenti provenienti da uno stesso server fisico, ottimizzando così l'uso delle risorse e riducendo i costi [24];
- sono separate dall'hardware fisico, infatti su ognuna di esse è possibile eseguire il sistema operativo più adatto all'applicazione che si vuole eseguire, anche se diverso dal server di origine, poiché funziona in maniera del tutto indipendente da esso. Ciò riflette il concetto alla base del cloud computing: ogni utente può utilizzare un servizio cloud pronto all'uso, che gira su diverse macchine virtuali, senza doversi minimamente preoccupare della complessità tecnica dell'infrastruttura hardware [24].

Questo approccio consente a più utenti non correlati di estrarre potenza di calcolo simultaneamente a partire dallo stesso server [18]. Queste macchine virtuali vengono coordinate dall'hypervisor, cioè un software che fa sì che ricevano le giuste risorse di cui necessitano dal server di riferimento per poter eseguire le applicazioni, ed inoltre garantisce che ogni macchina virtuale funziona in maniera indipendente dalle altre. In particolare, esistono due tipi di hypervisor:

- Hypervisor di tipo 1 o Bare-metal: si installano direttamente sull'hardware, sostituendo il sistema operativo tradizionale e gestiscono in modo diretto le risorse fisiche;
- Hypervisor di tipo 2: si installano direttamente sopra un sistema operativo già esistente e per poter gestire le risorse hardware devono passare attraverso il sistema operativo host. [24]

2.4 Architettura del cloud computing

Il cloud computing viene distribuito tramite 4 livelli: a partire dall'hardware fisico si creano delle risorse virtuali e integrate che mettono a disposizione servizi per sviluppare delle applicazioni che poi possono essere utilizzate dall'utente finale. Più nel dettaglio [18]:

1. Livello fisico: costituisce l'infrastruttura hardware del cloud ed è composta da server (computer fisici che eseguono i calcoli sui dati, allocati all'interno dei data center); spazi di archiviazione (dispositivi che archiviano i dati, come hard disk, SSD, ecc.) e risorse di rete (dispositivi che permettono ai server di comunicare tra di loro e con l'esterno). Tutta questa struttura viene gestita dal provider del servizio cloud e non dall'utente che decide di utilizzare la piattaforma cloud;
2. Livello di virtualizzazione: grazie a software specializzati, chiamati hypervisor, le risorse del livello precedente vengono separate dall'hardware fisico e integrate in risorse unificate, per poi essere distribuite tramite ambienti virtuali, costruiti all'interno dei server fisici;

3. Livello piattaforma: fornisce un ambiente completo dove gli sviluppatori possono creare e sviluppare le applicazioni senza doversi preoccupare del hardware. Alcuni degli strumenti che mette a disposizione sono:

   - API: interfacce che consentono agli sviluppatori di integrare applicazioni e servizi;
   - Middleware: software che funge da collante tra sistemi diversi per facilitare l'interazione tra applicazioni e dati;
   - Servizi di gestione: strumenti che monitorano e aggiornano le applicazioni;
   - Database as a Service: database che gli sviluppatori possono usare senza dover configurare l'infrastruttura.

4. Livello applicativo: è il livello utilizzato direttamente dagli utenti finali e contiene tutte le applicazioni cloud vere e proprie, ad esempio software di posta elettronica, strumenti di collaborazione, applicazioni di elaborazione testi, piattaforme di streaming, social network, ecc. In questo livello viene offerta un'interfaccia intuitiva, che si basa su API, ossia set di funzioni già pronte che consentono all'utente di interagire col cloud, accedere ai servizi, impostare delle risorse, programmare applicazioni ecc., nascondendo così tutta la complessità dell'infrastruttura sottostante [18]. L'esperienza dell'utente finale è di fondamentale importanza poiché su questo si basa il successo della piattaforma cloud. In particolare, gli aspetti chiavi che un utente deve ottenere nel momento in cui si interfaccia con l'ambiente cloud devono essere utilità, valore, credibilità e accessibilità.

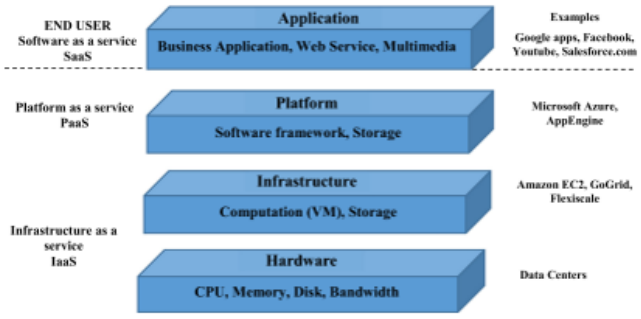

Figura 2.2. Architettura del cloud computing a livelli [33]

2.5 Servizi del cloud

I servizi offerti dal cloud possono essere organizzati in tre categorie fondamentali. Questa classificazione permette di distinguere quanto rimane sotto il controllo dell'utente e quanto, invece, viene gestito direttamente dal provider.

2.5.1 IaaS (Infrastructure as a service)

Detto anche hardware-as-a-service, è un modello di cloud computing in cui il provider mette a disposizione degli utenti tutte le risorse hardware di base, ossia risorse di calcolo, spazio di archiviazione, reti e connessioni. Tutta questa infrastruttura viene distribuita tramite macchine virtuali, che possono essere eliminate o modificate dinamicamente in base alle esigenze dell'utente.

L'utente ha quasi il pieno controllo sull'ambiente: non gestisce direttamente l'infrastruttura fisica ma può eseguire qualsiasi sistema operativo e installare applicazioni a sua scelta e configurare le risorse virtuali. L'accesso multi-dispositivo permette di utilizzare le VM da pc o smartphone senza procedure di setup particolarmente complesse, riducendo così tempi e costi. Inoltre, alcuni sistemi IaaS adottano algoritmi avanzati per ottimizzare il posizionamento delle VM, per ridurre sprechi di risorse e consumo energetico, e assicurare la continuità dei job anche in caso di guasti [27].

A differenza degli altri modelli in cui il sistema è pronto all'uso, qui l'utente può personalizzare tutto il sistema, senza dover gestire l'hardware fisico. Un esempio di IaaS è Amazon EC2, che fornisce server fisici virtuali, con specifiche definite dall'utente (memoria, OS, storage). [17][19][20][21]

2.5.2 PaaS (Platform as a service)

Nasce per supportare i limiti di IaaS, offrendo un sistema più strutturato in cui la scrittura del codice e la gestione delle applicazioni è nettamente migliore. In questo tipo di servizio il fornitore non si occupa solo dell'hardware ma anche dell'infrastruttura software, ossia sistema operativo, linguaggi di programmazione e strumenti di sviluppo per le applicazioni. Rappresenta un ponte intermedio tra hardware e applicazione, fornendo un ambiente completo, già pronto all'uso, con librerie, strumenti di sviluppo e ambienti di esecuzione dove gli sviluppatori possono creare, eseguire o ospitare applicazioni senza dover gestire direttamente l'ambiente di produzione.

Gli utenti possono accettare alcune limitazioni su quali tipi di software o linguaggi di programmazione utilizzare ottenendo in cambio una scalabilità automatica e una maggiore rapidità nello sviluppo.

Alcuni PaaS offrono sistemi avanzati di sicurezza e autenticazione, di bilanciamento del carico e disponibilità elevata dei servizi, anche in caso di guasti hardware. Altri strumenti, come le Content Delivery Network (CDN) integrate, memorizzando copie dei contenuti, permettono tempi di accesso ai contenuti molto più rapidi [27]. Esempi di PaaS sono Azure Data Factory e Google App Engine [17][19][20][21].

2.5.3 SaaS (Software as a service)

È un modello di servizio in cui il fornitore gestisce un software applicativo a cui gli utenti vi accedono tramite un'interfaccia intuitiva. Gli utenti non devono installare il software nel proprio computer ma accedono direttamente a un'applicazione integrata tramite Internet. Un esempio noto è Salesforce.com, che distribuisce soluzioni aziendali in abbonamento, come il sistema di gestione delle relazioni con i clienti (CRM). Anche molte applicazioni di uso quotidiano appartengono al mondo SaaS: servizi di posta elettronica, applicazioni per la dichiarazione dei redditi, software di elaborazione dei testi, ecc.

Un vantaggio fondamentale del SaaS è la rapidità di implementazione: mentre i sistemi tradizionali richiedono anni, un'applicazione SaaS può essere attivata in pochi mesi, senza costi iniziali per licenze software o hardware. Gli utenti possono utilizzare le applicazioni ovunque e in qualsiasi momento, basta avere una connessione Internet [27].

2.5.4 Servizi cloud compositi

Questi tre servizi possono comunicare tra di loro, ad esempio un servizio PaaS può usare le infrastrutture offerte da IaaS e allo stesso tempo fornire un ambiente SaaS per far girare le sue applicazioni.

Non tutti i servizi cloud rientrano nella definizione di IaaS, PaaS e SaaS. Ad esempio, App Store di Apple è un software che viene venduto tramite un modello pay-per-use ma la sua esecuzione avviene sui dispositivi degli utenti e non direttamente sui data center; nell'infrastruttura SaaS, invece, il software gira nel cloud e l'utente lo utilizza via Internet. Oppure i Servizi di eBook di Amazon, che combinano archiviazione, distribuzione globale dei contenuti, creando un ambiente cloud complesso che non può essere ricondotto a IaaS, PaaS o SaaS. Tutti questi servizi sono importanti quanto i modelli principali e vanno studiati come casi particolari del cloud computing. [17][19][20][21][22].

2.6 Modelli di distribuzione

Esistono cinque diversi modelli di distribuzione del cloud, che si sono sviluppati in momenti diversi: il cloud privato e pubblico negli anni 2000; il community cloud e il cloud ibrido

intorno al 2011 ed infine il multicloud tra il 2011 e il 2020 [37][39].

2.6.1 Cloud pubblico

E' il modello più comune di cloud computing. Qui le risorse vengono messe a disposizione non solo ad un singolo utente, ma tra tanti utenti diversi; inoltre l'infrastruttura informatica viene allocata solitamente presso la sede del provider. Esistono sia cloud pubblici gratuiti che privati, ad esempio Google offre spazio di archiviazione gratuito mentre EC2 Amazon è a pagamento. [20][21]

I principali vantaggi riguardano la semplicità di utilizzo, la scalabilità immediata e i costi ridotti. Esistono però anche dei rischi legati alla sicurezza e alla privacy, poiché i dati vengono memorizzati su server di terzi e non è sempre chiaro dove vengano fisicamente archiviati o con quali procedure di backup. [28]

Alcuni esempi di cloud pubblico sono: Amazon Elastic Compute Cloud, Blue Cloud di IBM, Google AppEngine e Windows Azure Services Platform.

2.6.2 Cloud privato

E' dedicato ad una singola organizzazione. Si tratta di un'architettura proprietaria, cioè gestita internamente da un fornitore dedicato, dove i servizi sono accessibili solo ad un numero ristretto di utenti, solitamente all'interno dell'organizzazione stessa. Ad esempio, una grande banca con molte filiali può collegare le proprie strutture informatiche per formare un cloud privato.

Questo modello offre gli stessi vantaggi del cloud pubblico, ma con un livello di sicurezza più alto, infatti viene scelto da organizzazioni che trattano dati sensibili. [20][21]

Inoltre, alcune analisi mostrano che, se integrato con infrastrutture già esistenti, può garantire anche una certa riduzione dei costi operativi [28]. Alcuni esempi di cloud privato sono eBay e HP CloudStart.

2.6.3 Community cloud

E' riservato ad una comunità ristretta (es. gruppo di università) che vuole condividere dati o risorse. L'infrastruttura può essere interna, cioè gestita direttamente dall'istituzione stessa che la possiede oppure esterna, cioè affidata ad un fornitore terzo.
Questo concetto è stato già introdotto da un altro tipo di infrastruttura informatica, ossia il grid computing, ma con l'aggiunta dell'erogazione on-demand, la scalabilità dinamica e un modello di servizio strutturato. [20][21]

Tra i vantaggi, vi è la possibilità di dividere i costi tra i partecipanti e di avere un'infrastruttura personalizzata in base alle esigenze comuni. Di contro, i costi rimangono superiori a quelli di un cloud pubblico e le risorse (banda e spazio) devono essere condivise fra tutti i membri [28].

2.6.4 Cloud ibrido

È un'infrastruttura che combina cloud pubblico, privato e community: un'organizzazione può, ad esempio, scegliere di archiviare i dati sensibili in un cloud privato e quelli meno critici in un cloud pubblico [20][21].

Questo modello permette di sfruttare la sicurezza del privato insieme alla flessibilità e convenienza del pubblico. Tra i principali vantaggi troviamo la possibilità di ottimizzare i costi in base al ciclo di vita delle applicazioni, usare il cloud pubblico per test e sviluppo, e il cloud privato per la produzione. Consente inoltre il cosiddetto "cloud bursting", cioè la possibilità di spostare temporaneamente carichi di lavoro sul cloud pubblico in caso di picchi di domanda [28].

Lo svantaggio principale è la maggiore complessità gestionale: integrare più ambienti cloud richiede strumenti avanzati di controllo e pone nuove sfide per la sicurezza, soprattutto nella gestione delle identità e della cifratura dei dati [28].

2.6.5 Multicloud

Esso si basa sull'utilizzo combinato di servizi offerti da più provider cloud, pubblici o privati, con l'obiettivo di evitare la dipendenza da un singolo fornitore e di sfruttare i punti di forza di ciascuna piattaforma. Il multicloud è nato in un secondo momento, rispetto agli altri modelli di distribuzione, in risposta ad una richiesta di maggior libertà di scelta e di riduzione del rischio di vendor lock-in.

Nonostante l'aumento di complessità derivante dall'utilizzo di più piattaforme insieme, il multicloud offre importanti vantaggi: maggiore flessibilità nella scelta dei servizi, riduzione dei costi e aumento delle prestazioni poiché i carichi di lavoro vengono allocati sul provider più adatto, e una maggiore resilienza grazie alla disponibilità di infrastrutture alternative in caso di malfunzionamenti. [38]

2.7 Proprietà e vantaggi del Cloud Computing

- Convenienza: uno degli aspetti distintivi del cloud computing è la riduzione dei costi di capitale, di manodopera e dei costi energetici [20]. Prima dell'introduzione del

cloud, nei modelli tradizionali le organizzazioni dovevano investire ingenti somme di denaro nelle infrastrutture informatiche: innanzitutto dovevano acquistare e installare tutti i server in anticipo e poi occuparsi della loro manutenzione nel tempo, anche se non venivano utilizzati al massimo delle loro capacità. Con l'avvento del cloud computing i costi di capitale e di manodopera si riducono notevolmente poiché le aziende che decidono di utilizzare una piattaforma cloud non devono più acquistare né l'infrastruttura hardware né le licenze software ed inoltre non devono più reclutare un team di esperti IT per gestire server e reti: è il fornitore del servizio cloud che deve occuparsi di questo investimento. Questo approccio è particolarmente utile quando l'attività da svolgere tramite l'utilizzo del cloud computing è una tantum, poiché si evitano spese ingenti per risorse che potrebbero rimanere inutilizzate. Inoltre, le risorse vengono messe a disposizione solo quando servono e si paga solo quello che viene effettivamente consumato. Questo tipo di modello di pagamento si chiama pay-per-use, ed il prezzo viene misurato su base oraria, al minuto o addirittura al secondo [22]. Ciò porta ad ulteriori vantaggi: time-to-market breve, infatti si può mettere in funzione un'applicazione senza aspettare mesi di installazione e configurazione e un ritorno sull'investimento rapido, poiché si investe meno all'inizio ma si guadagna rapidamente. Infine, esternalizzando l'IT, l'azienda non solo taglia i costi di acquisto e di gestione dei server ma anche i costi energetici dell'hardware, che rappresentano fino al 30-50% dei costi operativi. I data center, infatti, devono avere sistemi di alimentazione, raffreddamento, sicurezza ecc., quindi richiedono un'infrastruttura complessa che comporta spese ingenti. [20]

- Tolleranza ai guasti: la caratteristica chiave del cloud computing è il basso accoppiamento e l'indipendenza tra le diverse transazioni. Nei sistemi di calcolo parallelo tradizionali, anche un piccolo errore potrebbe compromettere intere elaborazioni.
  Nel cloud, invece, le operazioni, svolte da utenti non correlati, sono del tutto indipendenti: in caso di guasti non si genera un effetto a catena tale per cui se una transazione fallisce, questa blocca le altre. Di fatto, il cloud computing dimostra di avere un'alta tolleranza ai guasti.
  I guasti possono avvenire in 4 aree principali:

  - Provider-inner: problemi interni al provider, risolti tramite sistemi di backup o ridondanza;
  - Provider-across: problemi tra più provider, gestiti tramite annullamento o reindirizzamento delle transazioni;
  - Provider-user: problemi tra utente e provider, affrontati con algoritmi di tolleranza ai guasti bizantini;
  - User-across: problemi tra utenti (es. condivisione di risorse critiche in modo non sicuro).

Tutti questi guasti non sono così catastrofici, poiché esistono metodi consolidati in grado di ripristinare subito il sistema, garantendo così una forte capacità di resistere ai guasti [22].

- Flessibilità: le aziende spesso presentano picchi di carico temporanei, ad esempio durante campagne di vendita o in ore specifiche della giornata, e ciò richiede più risorse per non bloccare le attività. Nei sistemi tradizionali, per gestire questa situazione, le organizzazioni devono acquistare una quantità di risorse superiore a quella realmente necessaria, correndo però il rischio di lasciarle inutilizzate per lunghi periodi.
  Con il cloud, invece, la quantità di risorse messe a disposizione si adatta automaticamente alla domanda in tempo reale: quando si verifica un picco di utilizzo le risorse aumentano immediatamente, mentre nei momenti di minore attività vengono ridotte. Questa capacità di ridimensionare dinamicamente l'infrastruttura rende il cloud particolarmente adatto a scenari in cui la domanda è variabile e difficile da prevedere. Inoltre, l'elasticità non solo evita sprechi, ma garantisce anche continuità operativa. Tale approccio si rivela molto utile per le startup, che possono scalare progressivamente le risorse in base alla crescita del loro business, senza doversi preoccupare di pianificare in anticipo investimenti eccessivi.

- Disponibilità: prima che un servizio cloud venga reso disponibile deve essere stipulato un contratto tra il provider del servizio e il cliente: il Service Level Agreement (SLA). Questo documento definisce i parametri che devono essere garantiti dal fornitore per offrire un'esperienza affidabile.
  Alcuni esempi:

  – Disponibilità del servizio: percentuale di tempo in cui un servizio cloud deve rimanere attivo, solitamente 99,99% (circa 5 minuti di inattività tollerata ogni mese);
  – Tempo massimo di risposta: tempo massimo che deve intercorrere tra la segnalazione di un problema e la risposta del fornitore;
  – Garanzia di sicurezza: protezione dei dati da accessi non autorizzati, perdite o modifiche senza consenso;
  – Frequenza di backup: intervalli di tempo tra copie di sicurezza dei dati;
  – Regole di fatturazione: metodo e frequenza di pagamento;
  – Notifica di eventuali incidenti rilevanti: comunicazione tempestiva di guasti o attacchi informatici;
  – Politica di controllo in fase di esecuzione: modalità con cui il fornitore monitora il servizio mentre è in uso;
  – Conservazione dei dati dopo la cessazione del contratto: periodo di mantenimento prima dell'eliminazione definitiva;

  - Piano di recupero in caso di emergenza: procedure per il ripristino del servizio;
  - Modalità di risoluzione delle controversie: regole per la gestione dei conflitti tra cliente e fornitore.

- Affidabilità: un aspetto molto importante riguarda il rischio di downtime, ossia i periodi in cui il servizio non è disponibile. Per garantire stabilità nel tempo, l'infrastruttura cloud deve disporre di componenti ridondanti (server, dischi, connessioni di rete) che subentrano in caso di guasto. Inoltre, per prevenire la perdita di dati, le informazioni devono essere replicate su più server. Il requisito QoS (Quality of Service)non punta solo sulle prestazioni o sulla disponibilità, ma soprattutto sulla protezione dei dati e del codice applicativo. Questa attenzione si traduce in soluzioni software e hardware avanzate, come sistemi self-healing, capaci di migrare un'applicazione da un server guasto a uno funzionante. Tutti questi aspetti rendono il cloud un'infrastruttura robusta e affidabile, in grado di reagire a eventi critici di qualsiasi natura [18].

2.8 Sfide del cloud computing

Nonostante il cloud computing presenti numerosi vantaggi e sia oggi considerato una soluzione indispensabile per le imprese, il suo utilizzo porta con sé anche diverse sfide che non devono essere sottovalutate. Alcuni aspetti riguardano fattori tecnici e organizzativi, come la dipendenza da una connessione stabile, la definizione accurata degli SLA e la gestione ottimale delle risorse. Altri invece toccano questioni più delicate, come il rischio di vendor lock-in, le difficoltà nel garantire pienamente le proprietà ACID nelle applicazioni transazionali e i problemi legati alla protezione dei dati. Nei punti seguenti vengono analizzate nel dettaglio le principali criticità del cloud computing.

- Uno dei fattori essenziali del cloud computing è disporre di una connessione Internet stabile. Infatti, in caso di interruzioni, il servizio diventa inutilizzabile e quindi non è più possibile accedere ai dati e alle applicazioni cloud. Per evitare di incorrere in questo tipo di problemi, è utile disporre di un collegamento secondario pronto a subentrare in caso di guasto di quello principale, riducendo così il rischio di downtime. Ad esempio, se la connessione è in fibra ottica, quella secondaria potrebbe essere una connessione WiMAX [21].

- Prima di affidarsi al cloud computing, le aziende si accordano col fornitore stabilendo i parametri di qualità che il servizio cloud deve offrire (SLA). Molto spesso però questo contratto non viene definito in maniera chiara, per mancanza di competenze oppure per eccessiva fiducia nei confronti del fornitore. SLA poco chiari possono portare le aziende a riscontrare difficoltà nella gestione di malfunzionamenti, con ripercussioni sul proprio business.

- I dati vengono memorizzati in un sistema di archiviazione condiviso, che comporta il rischio che utenti non autorizzati, addirittura malintenzionati, possano alterare o rubare i dati. Per proteggersi da questo pericolo vengono adottate tecniche di crittografia, tramite le quali tutti i dati che devono essere trasferiti o salvati vengono cifrati. In questo modo chiunque acceda senza permesso a delle informazioni private non può leggerle o modificarle senza la chiave di decodifica. È importante stabilire se i dati personali possono essere trasferiti anche in paesi al di fuori dell'Unione Europea, dove possono verificarsi violazioni delle leggi europee (GDPR). Ad esempio nel Patriot Act statunitense vi sono leggi meno severe che riguardano la protezione dei dati: le autorità governative possono accedere a tutti i dati archiviati indipendentemente da dove si trovano fisicamente, senza il consenso dell'utente o dell'azienda [20]. A volte però i problemi legati alla sicurezza dei dati si verificano internamente all'azienda, magari a causa di disattenzione dei dipendenti.

- La qualità del servizio cloud potrebbe peggiorare nel tempo oppure i prezzi potrebbero aumentare, con il rischio di interrompere completamente l'attività e dover migrare tutte le funzionalità in un'altra piattaforma cloud. Questo è un processo alquanto complesso poiché ogni infrastruttura cloud non è standardizzata: ogni provider può utilizzare formati e interfacce diverse. Questo fenomeno è noto come vendor lock-in e rappresenta un rischio per le aziende poiché sono vincolate ad un fornitore in condizioni sfavorevoli. Per ridurre questo rischio è importante valutare con attenzione quale provider scegliere e soprattutto verificare che il contratto preveda clausole in caso di rescissione, in modo che venga adottata una migrazione rapida e poco costosa [21]. Sempre più aziende affrontano il problema adottando strategie multi-cloud, che distribuiscono i carichi di lavoro su più provider evitando di dipendere da uno solo [32].

- Nelle applicazioni che gestiscono i dati transazionali è importante che vengano rispettate le proprietà ACID (Atomicità, Coerenza, Isolamento, Durabilità), ossia che ogni operazione venga eseguita in modo affidabile. Nei sistemi di cloud computing questo è un requisito non facile da soddisfare, poiché vengono gestiti grandi volumi di dati replicati costantemente in server diversi. Per questo motivo il cloud è più adatto ad applicazioni di gestione di dati analitici dove i dati vengono aggiornati raramente.

- Il cloud computing mette a disposizione tantissimi tipi di risorse, che non è facile distribuire in maniera ottimale a tutti gli utenti poiché ognuno di essi ha esigenze diverse. In particolare, se la quantità di risorse a disposizione è limitata, può nascere competizione, con un impatto sia sui costi sia sulle prestazioni. È importante, quindi, garantire un equilibrio tra questi due aspetti per evitare che i clienti possano passare ad un altro fornitore. Per questo motivo è utile definire uno SLA chiaro che deve specificare non solo la qualità del servizio e le penalità in caso di disservizi, ma anche

le regole di allocazione delle risorse. Oggi, per ridurre i rischi legati a sicurezza e concorrenza nell'uso delle risorse, si stanno diffondendo modelli come Zero Trust, che si basa su controlli costanti e sull'assegnazione di privilegi minimi [30].

2.9 Sicurezza nel cloud computing

Una piattaforma di cloud computing viene detta multi-tenant, poiché è un'infrastruttura centralizzata le cui risorse vengono condivise da più utenti non correlati [18]. Questo approccio consente di ridurre i costi di gestione e manutenzione, ma può comportare rischi legati alla sicurezza: da un lato, la presenza di un elevato numero di informazioni private, dall'altro la condivisione di risorse che può esporre al pericolo di accessi non autorizzati tra utenti diversi. È quindi fondamentale che una piattaforma cloud disponga di efficaci tecniche di protezione, come l'archiviazione dei dati di ogni utente in database separati per prevenire accessi indebiti, o meccanismi di replica per garantire la continuità operativa in caso di guasto [19]. In questo contesto, un ruolo centrale è svolto dal GDPR (General Data Protection Regulation), la normativa europea che stabilisce delle regole precise per il trattamento dei dati personali. Per i fornitori di servizi cloud significa rispettare diversi obblighi: in primo luogo, garantire che i dati siano conservati in Paesi che applicano la normativa europea, oppure adottare misure di protezione equivalenti nel caso di trasferimento verso Paesi terzi. Un altro punto fondamentale è l'isolamento dei dati tra utenti diversi, così da evitare che un malfunzionamento o un errore di configurazione permetta ad altri di accedere a informazioni riservate. [26]

Il GDPR richiede anche l'adozione di strumenti tecnici per ridurre i rischi: tra questi rientrano la creazione di backup periodici, l'uso di autenticazione a due fattori per limitare l'accesso non autorizzato, e il monitoraggio continuo delle attività per individuare eventuali anomalie o violazioni. Ma non si tratta solo di sicurezza tecnica: la normativa sottolinea anche la necessità di trasparenza e responsabilità. Ciò significa che i provider devono informare chiaramente gli utenti su come vengono utilizzati i loro dati e mettere a disposizione meccanismi concreti per esercitare i diritti previsti, come il diritto di accesso ai propri dati, di cancellazione e di portabilità verso altri servizi [26].

Se inizialmente la sicurezza del cloud era concentrata soprattutto sulla protezione dei data center e sulla crittografia, oggi lo scenario è molto più complesso: gli utenti accedono da dispositivi e luoghi diversi e le applicazioni sono distribuite su più cloud. Per rispondere a queste nuove esigenze si è affermato il modello di Zero Trust Security, che si basa sull'idea di non considerare sicuro un utente o un dispositivo solo perché opera all'interno della rete aziendale. In questo approccio, qualsiasi accesso deve essere verificato, autenticato e autorizzato, riducendo al minimo i privilegi concessi [29]. Lo Zero Trust si fonda su tre principi chiave: il least privilege, che limita le autorizzazioni allo stretto necessario;

la microsegmentazione delle reti, che riduce i danni in caso di violazioni e l'autenticazione continua, che non si ferma al login iniziale ma verifica costantemente l'identità dell'utente e il contesto di accesso. Questa filosofia risulta particolarmente efficace in ambienti multi-tenant, dove le risorse sono condivise e il rischio di accessi non autorizzati è maggiore [30].

Un ulteriore passo avanti è rappresentato dall'integrazione dell'intelligenza artificiale nei sistemi di difesa. Algoritmi di machine learning analizzano in tempo reale grandi quantità di dati per individuare comportamenti anomali che potrebbero segnalare un attacco in corso. In questo modo il cloud può reagire immediatamente, aumentando la resilienza complessiva delle infrastrutture [31].

Accanto a questi approcci innovativi rimangono essenziali pratiche consolidate come la crittografia dei dati in transito e a riposo e i sistemi di monitoraggio continuo. Nel complesso, la sicurezza nel cloud è oggi un requisito centrale, non più un aspetto accessorio: solo un approccio strutturato e dinamico consente di garantire protezione e affidabilità [2].

Capitolo 3

Microsoft Azure

Come abbiamo già più volte evidenziato, sempre più aziende scelgono di adottare il cloud computing per le proprie esigenze in ambito computazionale, di archiviazione e di networking. Una piattaforma cloud che ha riscosso parecchio successo a livello globale è Microsoft Azure. Grazie all'ampia gamma di servizi, alla scalabilità e alla flessibilità, Azure consente alle aziende di trarre vantaggi competitivi concreti.

Si tratta di una piattaforma cloud progettata da Microsoft, che aiuta le aziende a soddisfare le proprie esigenze aziendali. In particolare, tramite i suoi strumenti permette alle aziende di archiviare e analizzare dati, sviluppare applicazioni, eseguire attività di machine learning e molto altro, senza doversi preoccupare dell'infrastruttura hardware.

Dalla sua nascita, nel 2010, Azure si è evoluto rapidamente, ampliando i propri data center, introducendo nuove tecniche di virtualizzazione e nuove applicazioni per rispondere alla crescente domanda. [41][42]

Figura 3.1. Rete globale Microsoft: distribuzione di regioni e nodi edge [76]

3.1 Servizi di Azure

A seconda della modalità di erogazione, i servizi messi a disposizione da Azure possono classificarsi in Infrastructure as a Service (IaaS), Platform as a Service (PaaS) e Software as a Service (SaaS).

In particolare, i servizi che appartengono ad ognuna di queste 3 macrocategorie possono coprire diversi ambiti, tra cui Elaborazione, Archiviazione e Networking, ciascuno con caratteristiche specifiche per rispondere alle diverse esigenze aziendali.

3.1.1 Azure Compute

- Azure Virtual Machines (VMs): è un servizio IaaS che permette di creare macchine virtuali all'interno dell'infrastruttura cloud. In pratica, si tratta di ambienti completamente configurabili, in cui è possibile scegliere il sistema operativo desiderato (ad esempio Windows o Linux) ed eseguire applicazioni aziendali con lo stesso livello di controllo di un server fisico. Il vantaggio principale delle Azure Virtual Machines è la loro flessibilità: le risorse (CPU, RAM, memoria, GPU, spazio) possono essere aumentate o ridotte in qualsiasi momento in base alle esigenze aziendali. Inoltre, possono essere distribuite in diverse Availability Zones, ossia gruppi di data center indipendenti progettati per garantire continuità di servizio. Esiste anche un'estensione di questo servizio, chiamata Azure Virtual Machine Scale Sets, che permette di gestire in modo automatico più macchine virtuali identiche. Ad esempio, se un'applicazione riceve un numero troppo elevato di richieste, il sistema può avviare nuove macchine virtuali per distribuire il carico di lavoro e mantenere elevate le prestazioni [44].

- Azure Kubernetes Service (AKS): è un servizio cloud PaaS che permette di eseguire applicazioni containerizzate utilizzando Kubernetes, ovvero il sistema di orchestrazione dei container fornito da Azure. Quando parliamo di applicazioni containerizzate ci riferiamo a container che racchiudono al proprio interno tutto ciò che serve per eseguire quella determinata applicazione, ossia codice, librerie e dipendenze. In questo contesto, il ruolo di Kubernetes è quello di coordinare tutti i container in maniera automatizzata. In particolare, svolge i seguenti compiti:

  - per avviare un'applicazione containerizzata può distribuire i vari container su più macchine virtuali, per aumentare l'affidabilità e per migliorare le prestazioni;
  - distribuisce in maniera uniforme il carico di lavoro su più container in modo tale che nessuno di essi vada in sovraccarico;
  - aggiorna le vecchie versioni dei container senza interrompere l'attività dell'applicazione;

  - sostituisce in automatico una macchina virtuale in caso di guasto e sposta i container su una macchina virtuale funzionante.

  AKS è pensato per la gestione di applicazioni containerizzate complesse, che richiedono orchestrazione e alta resilienza [41].

- Azure App Service: è un servizio cloud di tipo PaaS, che permette di creare e ospitare applicazioni web senza preoccuparsi del server o della manutenzione tecnica. Supporta diversi linguaggi di programmazione, ma consente anche di utilizzare immagini Docker (modelli preconfigurati che includono applicazione e dipendenze) e di eseguirle come container (istanze attive e isolate di tali immagini), garantendo così la massima flessibilità nello sviluppo e nel rilascio delle applicazioni. Le sue principali funzionalità sono:

  - ridimensionamento automatico delle risorse in base al carico di lavoro richiesto;
  - distribuzione continua integrata con GitHub, Azure DevOps e altri strumenti, per aggiornamenti rapidi e sicuri delle applicazioni;
  - strumenti di monitoraggio e diagnostica integrati, per controllare errori e quindi intervenire per ottimizzare le applicazioni. [41]

- Azure Container Instances (ACI): è un servizio cloud PaaS che permette di eseguire un container in modo rapido, senza dover gestire l'infrastruttura sottostante. In particolare, si tratta di una soluzione utile se si vuole eseguire un'applicazione containerizzata in modo veloce, con piccoli carichi o esecuzioni temporanee. A differenza di AKS, ACI non fornisce funzionalità di orchestrazione complesse: è pensato per scenari semplici e immediati, dove conta più la velocità di esecuzione che la gestione su larga scala [41].

3.1.2 Azure Storage

- Azure Blob Storage: è un servizio ideale per l'archiviazione di dati non strutturati, come file, immagini, video, ecc. Infatti, la parola "Blob" significa "Binary Large Object" e indica file di qualsiasi tipo e dimensione. In particolare, supporta tre tipi di formati: Block Blob, per file di grandi dimensioni, come immagini e video; Append Blob, per dati che vanno aggiunti in sequenza, come log o file di monitoraggio; e infine Page Blob, per dati che richiedono un accesso rapido, come file di un disco virtuale. Inoltre, i dati possono essere archiviati in diversi livelli in base alla frequenza con cui vi si accede (Hot, Cool, Archive). [41]

- Azure File Storage e Azure Data Lake Storage: entrambi i servizi offrono modalità di archiviazione più strutturate rispetto a Blob Storage. Il primo consente di creare condivisioni di file nel cloud, accessibili come se fossero in una rete locale. In questo

modo i file non restano vincolati alla rete interna dell'azienda, ma possono essere gestiti e condivisi in modo sicuro anche da sedi diverse o da remoto. Il secondo, invece, è progettato per gestire enormi volumi di dati (strutturati e non strutturati) che devono essere sottoposti ad attività di analisi avanzata. [41]

- Azure SQL Database e Azure Open Source Database: questi servizi offrono database relazionali pensati per gestire transazioni online (OLTP) in modo rapido e sicuro. Azure SQL Database è la versione cloud di Microsoft SQL Server ed è un servizio completamente gestito: questo significa che Azure si occupa di tutte le attività tecniche come aggiornamenti, patch di sicurezza, backup e scalabilità, così le aziende devono solo concentrarsi sull'utilizzo del database. Azure Open Source Database mette invece a disposizione versioni gestite di motori molto diffusi come MySQL, PostgreSQL e MariaDB. In questo modo le imprese possono continuare a utilizzare strumenti open source familiari, ma con i vantaggi del cloud, come affidabilità, sicurezza integrata e gestione semplificata. [41]

- Azure Cosmos DB: si tratta di un database distribuito a livello globale che può gestire una quantità elevata di dati, offrendo bassa latenza e scalabilità. Supporta diversi modelli di dati, ossia documenti JSON, coppie chiave-valore, grafi e famiglie di colonne. È ideale per applicazioni distribuite in tutto il mondo, come servizi cloud-native, app mobili e IoT che richiedono accesso rapido e affidabile ai dati. [41]

3.1.3 Azure Networking

- Azure Virtual Network (VNet): le Virtual Network in Azure rappresentano una rete privata nel cloud, che permette di far comunicare tra loro le risorse Azure, come macchine virtuali, App Service, database, container, offrendo un controllo totale sul traffico della rete. All'interno di una VNet si possono creare delle sottoreti, cioè porzioni più piccole della rete principale, che servono a raggruppare le risorse per meglio gestire il traffico, ottenendo così un livello di sicurezza più alto. [41]

- Azure VPN e Express Route: si tratta di due soluzioni che consentono di gestire in maniera sicura la connessione tra la rete aziendale locale e la rete virtuale. In particolare, la prima soluzione lavora come un tunnel criptato per proteggere i dati mentre viaggiano dalla rete locale alla rete virtuale di Azure. Si tratta di un servizio economico ma con un livello di sicurezza e di prestazioni bassi poiché i dati vengono criptati su internet. Nella seconda soluzione, invece, i dati vengono trasmessi tramite una linea privata, garantendo così elevate prestazioni e un massimo livello di sicurezza, ma al tempo stesso prevede costi elevati. [41]

- Private Endpoint: è una soluzione che rende accessibile un servizio Azure solo tramite la rete privata dell'azienda (VNet), usando un indirizzo IP privato, riducendo così l'esposizione esterna e migliorando la sicurezza. È particolarmente utile per servizi

che gestiscono dati sensibili, come ad esempio Azure Storage, SQL Database o Key Vault, che quindi possono essere raggiunti in maniera sicura solo dalla rete aziendale. [41]

3.1.4 Azure Internet of Things

- Azure IoT Hub: è la piattaforma centrale di Azure per l'Internet of Things e consente di connettere milioni di dispositivi (sensori, macchine, oggetti connessi) distribuiti in scala mondiale con il cloud di Azure. Ogni dispositivo ha una propria identità e comunica in modo sicuro con la piattaforma, inviando dati di telemetria in tempo reale e ricevendo, a sua volta, comandi o aggiornamenti dal cloud. Oltre alla raccolta dati, IoT Hub permette di registrare nuovi dispositivi, monitorare il loro stato e modificarne le configurazioni da remoto. I dati raccolti possono poi essere inviati e analizzati tramite altri strumenti Azure come Stream Analytics, Functions o Machine Learning, per sottoporli ad attività di analisi predittiva e automazione. Rispetto ad altre soluzioni IoT, offre un livello superiore di controllo e flessibilità, ma richiede anche maggiori competenze tecniche per la gestione e l'integrazione. [41]

- Azure IoT Central: è una piattaforma IoT pronta all'uso che consente di distribuire applicazioni IoT in modo semplice e veloce, senza dover scrivere codice complesso. A differenza di IoT Hub, non richiede una gestione diretta dell'infrastruttura e dell'orchestrazione dei dispositivi, perché mette già a disposizione modelli preconfigurati per diversi scenari industriali (come produzione, logistica o smart building). Inoltre, fornisce dashboard interattive per monitorare in tempo reale lo stato dei dispositivi connessi e interfacce grafiche intuitive che permettono di creare regole, avvisi e flussi di lavoro personalizzati. Questo lo rende uno strumento accessibile anche a chi non è sviluppatore o non possiede competenze tecniche avanzate, ma vuole comunque implementare soluzioni IoT scalabili e sicure in tempi rapidi. [41]

3.1.5 Azure Big Data

- Azure Data Lake Storage (ADLS): è un sistema di archiviazione cloud progettato per conservare enormi quantità di dati grezzi, ossia sia strutturati che non strutturati, indipendentemente dal formato. La sua caratteristica principale è l'organizzazione gerarchica in cartelle e sottocartelle, che facilita la gestione dei dati. La sua funzione è quella di fungere da punto di raccolta centrale per i dati grezzi provenienti da fonti eterogenee, che in un secondo momento possono essere elaborati ed analizzati con strumenti di Big Data e intelligenza artificiale. [41]

- Azure Synapse Analytics: è una piattaforma centralizzata di analisi dei dati, che non svolge solo la funzione di archiviazione, ma è progettato principalmente per interrogarli ed elaborarli. Combina le funzionalità di un tradizionale data warehouse

con strumenti di Big Data, consentendo di scrivere query con linguaggi familiari come SQL, Spark o Python. Può attingere a dati provenienti da diverse fonti, come Azure Data Lake Storage o altri sistemi aziendali, ed elaborarli per ottenere insights significativi, utili per supportare decisioni strategiche basate sui dati. Inoltre, Synapse si integra con servizi come Power BI per creare report e dashboard interattivi e con Data Factory per collegare e trasformare i dati, consentendo di costruire flussi analitici completi dall'acquisizione fino alla visualizzazione. [41]

- Azure HDInsight: è un servizio cloud completamente gestito che consente di utilizzare i più diffusi framework open-source per l'elaborazione distribuita dei dati, come Apache Hadoop, Spark, Hive, Kafka e molti altri. Grazie a questa piattaforma, le aziende possono eseguire processi complessi di analisi su Big Data, costruire modelli di machine learning e creare pipeline di dati scalabili senza doversi occupare della gestione dell'infrastruttura sottostante. Ciò non significa solo evitare la gestione di server fisici (come accade in generale nel cloud), ma soprattutto non doversi occupare della complessa configurazione, manutenzione e aggiornamento dei cluster Big Data e dei framework open-source, attività che richiedono competenze tecniche avanzate. [41]

- Azure Data Factory (ADF): è lo strumento di integrazione dei dati di Azure, che consente di creare e gestire pipeline per spostare, trasformare e combinare dati provenienti da diverse fonti. Con ADF è possibile automatizzare i processi di estrazione, trasformazione e caricamento (ETL) e rendere disponibili i dati in una forma coerente per le successive analisi. La piattaforma supporta connettori verso un'ampia varietà di sorgenti, sia cloud che on-premises, ed è quindi molto utilizzata per creare flussi di dati ibridi e complessi. [41]

- Azure Databricks: è una piattaforma di analisi avanzata basata su Apache Spark che permette di elaborare grandi volumi di dati provenienti da fonti diverse, pulirli, trasformarli e prepararli per successive analisi. Consente anche di sviluppare modelli di machine learning e applicazioni di intelligenza artificiale sfruttando la potenza del cloud. Databricks si integra con altri servizi Azure, come Data Lake Storage per l'archiviazione e Synapse Analytics per l'analisi, creando pipeline complete dalla raccolta dei dati fino ai modelli predittivi e alle dashboard. Supporta più linguaggi di programmazione (Python, R, Scala, SQL) ed è uno strumento flessibile per team con competenze diverse. Grazie a queste caratteristiche, rappresenta uno degli strumenti chiave per l'approccio del Data Lakehouse, che unisce in un unico ambiente la gestione dei dati grezzi e le capacità analitiche avanzate. [42]

3.1.6 Azure Artificial Intelligence

- Azure Machine Learning (Azure ML): è una piattaforma cloud che permette di creare modelli di machine learning, attraverso strumenti che prevedono l'utilizzo di linguaggi di programmazione o strumenti visivi no-code, come ad esempio Azure Machine Learning Studio, che permette di creare pipeline di machine learning in modo grafico. Inoltre, fornisce modelli già pronti (es. riconoscimento immagini, analisi del testo, ecc.), che possono essere ri-addestrati successivamente per adattarsi a scenari più specifici. Azure ML non si limita alla creazione dei modelli, ma gestisce tutto il ciclo di vita, dalla sperimentazione alla messa in produzione e integra concetti di MLOps (Machine Learning Operations) per mantenere i modelli aggiornati. [41]

- Servizi cognitivi: si tratta di un insieme di API e strumenti cloud che aggiungono funzionalità di intelligenza artificiale avanzata alle applicazioni, senza dover sviluppare da zero algoritmi complessi. Le API consentono alle applicazioni di comunicare direttamente con i servizi di Azure per eseguire compiti intelligenti, ad esempio per riconoscere volti, comprendere ed analizzare testi, generare riassunti e traduzioni automatiche, trasformare la voce in testo, ecc. Si tratta, quindi, di soluzioni immediate, adatte quando serve applicare funzioni comuni. [41]

- Azure OpenAI Service: è un servizio che permette di usare modelli di intelligenza artificiale generativa molto più avanzati, sviluppati da OpenAI e integrati in Azure. Include GPT-4 e GPT-3.5 per generare e comprendere testi, Codex per scrivere codice, DALL · E per creare immagini da descrizioni testuali e Whisper per trascrivere e tradurre audio. Questi modelli sono più flessibili e creativi rispetto ai Servizi cognitivi e possono essere adattati a tanti scenari diversi, come assistenti virtuali, analisi documentale o generazione di contenuti, con i vantaggi di sicurezza e scalabilità offerti da Azure. [41]

- Azure Bot Services: permettono di costruire chatbot intelligenti in grado di comprendere e rispondere al linguaggio naturale. I bot possono essere integrati in diversi canali di comunicazione, come siti web, applicazioni, Microsoft Teams o WhatsApp, e possono sfruttare i Servizi cognitivi o Azure OpenAI per migliorare la comprensione del linguaggio e la qualità delle risposte. [41]

3.1.7 Azure DevOps

- Azure Boards: fornisce strumenti per gestire i progetti in modo agile, aiutando a tenere traccia delle attività da fare (backlog), pianificare i periodi di lavoro (sprint), assegnare i compiti ai membri del team e controllare i progressi tramite report. In questo modo il lavoro del gruppo risulta più organizzato e chiaro. [41]

- Azure Repos: offre repository Git e TFVC per salvare il codice e tenerne traccia nel tempo. In questo modo più sviluppatori possono lavorare insieme sullo stesso progetto, controllando facilmente le modifiche e mantenendo il lavoro sicuro e organizzato. [41]

- Azure Pipelines: permette di automatizzare la creazione, il test e la pubblicazione del software tramite pipeline CI/CD. In questo modo i team possono rilasciare nuove versioni delle applicazioni in modo più veloce e affidabile, senza dover fare ogni passaggio manualmente. [41]

- Azure Test Plans: mette a disposizione strumenti per eseguire test manuali e automatizzati, oltre che per pianificare campagne di test strutturate. L'obiettivo è assicurare la qualità del software lungo tutto il ciclo di sviluppo. [41]

- Azure Artifacts: permette di gestire e condividere pacchetti di codice, come NuGet, npm o Maven, tra i membri del team. Questo aiuta a usare sempre le stesse versioni delle librerie e a riutilizzare più facilmente i componenti già sviluppati. [41]

3.1.8 Azure Disaster Recovery

- Azure Site Recovery: consente di replicare in modo continuo le macchine virtuali in un ambiente secondario. In caso di guasti o interruzioni nei data center principali, i servizi possono essere riattivati rapidamente nell'ambiente di backup, garantendo la continuità operativa. [43]

- Azure Backup: permette di salvare copie di file, cartelle e database in contenitori di backup sicuri. Queste copie possono essere utilizzate per recuperare i dati in caso di guasti hardware, cancellazioni accidentali o attacchi informatici. [43]

3.1.9 Azure Migration Services

- Azure Migrate: si tratta di una piattaforma centralizzata che consente di pianificare ed eseguire la migrazione di server, macchine virtuali, applicazioni e database dal mainframe al cloud. Attraverso strumenti di discovery e di assessment è possibile analizzare l'infrastruttura esistente, stimando i costi di esercizio su Azure e individuando eventuali incompatibilità prima del trasferimento. Inoltre, Azure Migrate permette di visualizzare le dipendenze tra i diversi workload, così da pianificare una migrazione a ondate (waves) e ridurre il rischio di interruzioni improvvise dei servizi. Una volta completata la fase di valutazione, la piattaforma consente di avviare la migrazione vera e propria. [41][42]

- Azure Database Migration Service (DMS): è uno strumento specifico per la migrazione dei database, che supporta motori come SQL Server, MySQL, PostgreSQL e

MariaDB, e permette di trasferire i dati verso le soluzioni equivalenti gestite in Azure, come Azure SQL Database o Azure SQL Managed Instance. Una delle principali potenzialità del DMS è la possibilità di eseguire la migrazione in modalità online, mantenendo sincronizzati il database sorgente e quello di destinazione fino al momento del cutover, riducendo così al minimo il downtime per le applicazioni aziendali critiche. Al contrario, la modalità offline è pensata per scenari meno sensibili, dove è accettabile un periodo di inattività più lungo. Grazie all'integrazione con strumenti di analisi preventiva, come il Data Migration Assistant, le aziende possono ricevere indicazioni chiare su eventuali funzionalità non compatibili o su ottimizzazioni da eseguire prima del passaggio. In questo modo, la migrazione dei database diventa più sicura, scalabile e allineata alle esigenze operative. [42][43]

3.2 Punti di forza e confronto competitivo

Microsoft Azure si distingue per una serie di caratteristiche che gli permettono di distinguersi rispetto agli altri due principali competitor nel mercato del cloud, ovvero Google Cloud Platform e Amazon Web Services.

Figura 3.2. Loghi dei principali provider cloud: Microsoft Azure, Amazon Web Services (AWS) e Google Cloud Platform (GCP) [77]

3.2.1 Copertura globale

Tra queste caratteristiche, la copertura globale della rete di data center è un aspetto di rilievo, infatti permette di distribuire i dati in oltre 70 regioni del mondo [44]. In particolare, ogni organizzazione può scegliere in quale area geografica conservare i propri dati, così da garantire il rispetto delle normative che ritiene più adatte alle proprie esigenze. Ad esempio, in Europa esistono normative molto severe come il GDPR, negli Stati Uniti l'HIPAA per i dati sanitari o il CCPA in California. Questo aspetto consente non solo di assicurare la conformità legale, ma anche di migliorare le prestazioni, poiché i dati possono essere collocati più vicino agli utenti finali, riducendo così la latenza.

Gli altri due competitor offrono anch'essi la possibilità di scegliere dove conservare i propri dati ma presentano una copertura più limitata, sia in termini di numero di regioni disponibili sia di distribuzione geografica equilibrata: AWS dispone di 37 regioni e 117 Availability Zones; invece Google Cloud conta 42 regioni e 127 Availability Zones [47][48].

3.2.2 Integrazione con l'ecosistema Microsoft

Un altro aspetto distintivo di Azure è la sua integrazione nativa con l'ecosistema Microsoft. La maggior parte delle imprese utilizza da tempo soluzioni Microsoft come Windows Server, Active Directory, SQL Server o la suite Microsoft 365, che non hanno bisogno di stravolgere la propria infrastruttura per passare al cloud. Essendo servizi che già fanno parte dell'ecosistema Microsoft, una volta trasferiti su Azure, funzionano in modo naturale senza richiedere grandi modifiche o interventi complessi. Per questi motivi, la migrazione al cloud risulta più rapida e semplice, con una riduzione sia dei costi legati al trasferimento sia di quelli relativi alla formazione del personale [41].

Bisogna però precisare che il vantaggio principale di Azure non dipende dal fatto che i servizi Microsoft siano più compatibili con il cloud rispetto ad AWS o Google Cloud, ma dal fatto che le imprese li utilizzano da decenni. Al contrario, i servizi nativi di Amazon e Google non hanno avuto la stessa diffusione nelle aziende, di conseguenza pur supportando integrazioni con software open source e soluzioni di terze parti, non possono offrire lo stesso livello di immediatezza che caratterizza Azure [45].

3.2.3 Hybrid Cloud e continuità dei sistemi on-premises

Un ulteriore punto di forza di Azure è la sua capacità di supportare l'hybrid cloud. A differenza di altri competitor che adottano quasi esclusivamente modelli full-cloud, Microsoft ha sviluppato strumenti come Azure Arc e Azure Stack, che consentono di collegare i sistemi che si trovano ancora nei data center dell'azienda (on-premises) con i servizi cloud, facendoli funzionare in modo integrato. In questo modo, anche se i sistemi appartengono ad ambienti diversi, vengono comunque gestiti con gli stessi strumenti di controllo, le stesse regole di sicurezza e procedure di governance. Questo approccio è particolarmente utile quando si vuole intraprendere un progetto di migrazione da mainframe, in quanto questi sistemi gestiscono dei processi critichi che non possono essere fermati all'improvviso. Invece, grazie all'approccio del modello ibrido offerto da Azure, le applicazioni possono essere spostate gradualmente proprio perché risultano già connesse all'interno dell'ambiente cloud [42].

AWS offre delle soluzioni per poter estendere i propri servizi cloud ai data center aziendali, ma con un livello di integrazione molto più limitato, poiché impone il proprio modello ai

sistemi on-premises. Google Cloud, invece, è più flessibile infatti non vincola le aziende alle proprie regole, ma essendo fortemente orientato alle tecnologie moderne del cloud-native (container e Kubernetes) può gestire solo le applicazioni già modernizzate e non è pensato per le aziende che usano sistemi legacy tradizionali [46]

3.2.4 Sicurezza e compliance normativa

Azure si distingue per la particolare attenzione verso sicurezza e alla compliance normativa. Azure dispone di data center certificati secondo standard internazionali come ISO (norme per la qualità e la sicurezza dei sistemi informativi), GDPR (regolamento europeo per la protezione dei dati personali), HIPAA (normativa statunitense sulla tutela dei dati sanitari) e SOC (controlli di sicurezza e affidabilità sui sistemi IT). In questo modo le aziende possono rispettare i requisiti normativi senza dover creare server o sistemi di sicurezza aggiuntivi, poiché queste garanzie sono già integrate all'interno della piattaforma cloud. Infatti, Azure dispone di un sistema di sicurezza centralizzato fornito tramite strumenti nativi: Microsoft Defender for Cloud, che controlla costantemente l'infrastruttura cloud per verificare minacce o rischi di attacco; sistemi di prevenzione delle vulnerabilità, che individuano eventuali punti deboli nei sistemi che potrebbero essere sfruttati dagli hacker ed infine meccanismi di crittografia avanzata dei dati sia a riposo che in transito [43].

AWS offre un livello di sicurezza molto avanzato e un'ampia gamma di certificazioni, però per controllare che le norme vengano rispettate è necessario utilizzare strumenti esterni aggiuntivi. Google Cloud propone soluzioni robuste in termini di sicurezza, ma presenta un numero inferiore di certificazioni riconosciute a livello globale, risultando così meno adatto in settori in cui le certificazioni sono un requisito essenziale, come ad esempio ospedali, banche, ecc. [47]

Capitolo 4

Migrazione dei mainframe al cloud: approcci e roadmap

I mainframe esistono da decenni e vengono tuttora impiegati in numerosi settori, come quello bancario, sanitario, governativo, grazie alla loro affidabilità e alla capacità di gestire un numero elevato di transazioni simultanee. Tuttavia, come già evidenziato nel capitolo dedicato, presentano diversi limiti che rallentano l'innovazione: elevati costi di manutenzione, scarsa flessibilità alle modifiche, difficoltà di integrazione con altri sistemi e carenza di personale specializzato.

Per queste ragioni la maggior parte delle organizzazioni ha deciso di migrare verso piattaforme cloud, trasferendo applicazioni, dati e carichi di lavoro dai sistemi mainframe, poiché offre tutti i vantaggi che al mainframe mancano, ossia elevata scalabilità, costi ridotti e migliori prestazioni.

Secondo un'indagine condotta da Accenture nel 2022 sulla modernizzazione dei mainframe, il 93 % delle aziende che utilizzano il mainframe sta già migrando o sta valutando di spostare le proprie applicazioni sul cloud [53].

Bisogna precisare però che il mainframe non è destinato a scomparire: basti pensare che ne viene rilasciato uno nuovo ogni due o tre anni da IBM, con livelli di sicurezza e potenze di calcolo sempre più alte. Ad esempio, IBM z16 integra un processore di ultima generazione ad alte prestazioni, acceleratori di intelligenza artificiale, funzionalità avanzate di sostenibilità e supporto alla crittografia quantistica [53].

Nonostante questi sviluppi, il mainframe resta un sistema poco flessibile e oneroso da mantenere. È per questo che sempre più imprese guardano al cloud come a una strada necessaria per innovare, semplificare i processi e aprirsi a nuove opportunità tecnologiche.

4.1 Vantaggi della migrazione

La migrazione dal mainframe al cloud non rappresenta soltanto un cambiamento tecnologico, ma porta con sé una serie di benefici concreti per le aziende, che descriviamo di seguito.

- I mainframe utilizzano linguaggi datati come COBOL, PL/I e FORTRAN, che sono complessi da migliorare. Con le applicazioni sul cloud, invece, le aziende possono adottare linguaggi e framework attuali, più adatti a supportare processi di sviluppo agili. Ciò rende i sistemi più predisposti alle innovazioni future, attraendo così giovani professionisti, che tendono a preferire ambienti tecnologici contemporanei.
- Nel mainframe è necessario acquistare in anticipo l'infrastruttura hardware, col rischio di non utilizzarla al massimo delle sue capacità. Invece, col cloud, è possibile aumentare le risorse in tempo reale, in base alle esigenze degli utenti, evitando sprechi.
- Nel mainframe l'attivazione di nuove applicazioni richiede lunghi processi di configurazione; invece, nel cloud è possibile creare nuovi ambienti con pochi click, tramite interfacce self-service. In questo modo, l'azienda può reagire rapidamente alle nuove esigenze di mercato.
- Nel momento in cui le applicazioni vengono ospitate nel cloud, l'infrastruttura viene gestita dal provider, che si occupa di manutenzione e aggiornamenti. L'azienda può concentrarsi sulle proprie attività e non deve più dedicarsi alla gestione tecnica quotidiana dei propri sistemi legacy [52].

Fatte queste premesse, migrare i propri sistemi mainframe verso soluzioni cloud sembra essere la decisione migliore. Tuttavia, l'esperienza ha mostrato che molte aziende hanno incontrato difficoltà nella migrazione e non sempre hanno ottenuto i benefici attesi, non a causa della piattaforma cloud in sé, ma principalmente per le modalità con cui il processo è stato gestito.

Prima di analizzare nel dettaglio tutte le fasi di cui è composto questo percorso, è utile soffermarsi sulle principali sfide che possono emergere durante l'esecuzione della migrazione. Comprendere queste criticità aiuta a mettere in atto strategie adeguate per affrontare la transizione nel modo più efficace possibile.

4.2 Sfide della migrazione

Le applicazioni all'interno di un sistema mainframe sono state sviluppate decenni fa e, nel tempo, hanno incorporato una molteplicità di regole e logiche, strettamente intrecciate

tra di loro. Questa struttura fortemente integrata rende difficile isolarne le componenti e adattarle ad un ambiente cloud. In particolare:

- Le applicazioni mainframe vengono scritte in linguaggi vecchi e soprattutto molto diversi da quelli moderni utilizzati dalle nuove generazioni di sviluppatori (es. Java, Python, ecc.). Trattandosi di milioni di righe di codice scritte da team diversi e in tempi diversi, senza una riscrittura da zero diventa complicato convertirlo in un linguaggio più moderno, poiché non è necessario soltanto tradurre la sintassi, ma anche reinterpretare le logiche di business [54];

- Le applicazioni mainframe rappresentano un unico grande blocco di software, in cui ogni modulo dipende fortemente dagli altri; di conseguenza se ne viene modificato uno, questo può avere degli effetti imprevisti sugli altri. Nel cloud, invece, i microservizi sono programmi indipendenti, che possono essere modificati o sostituiti senza compromettere l'intero sistema. Quando si vuole migrare un'applicazione dal mainframe al cloud, non si può spezzettarla in moduli in quanto strettamente interconnessi; è quindi necessario riscrivere l'intero codice da capo per renderlo compatibile con l'ambiente cloud [54]. Inoltre, non esiste una documentazione che riflette l'evoluzione del codice e ciò porta all'esigenza di fare reverse engineering, ossia analizzare il codice sorgente, database, log del sistema in esecuzione per ricostruire le regole di business e le interfacce tra i moduli [54];

- Quando si effettua una migrazione si spostano grandi volumi di informazioni critiche, quindi è importante verificare che i dati non vengano alterati o persi durante il trasferimento, restino allineati tra le diverse tabelle e siano soggetti a leggi sulla privacy e sicurezza. Inoltre, la migrazione deve avvenire senza bloccare i sistemi aziendali che usano quei dati;

- I mainframe, essendo sistemi chiusi, risultano facilmente controllabili e protetti da accessi autorizzati. Passare ad una piattaforma cloud significa introdurre nuovi rischi legati alla sicurezza: i dati diventano accessibili tramite Internet e possono essere condivisi da più utenti contemporaneamente. È quindi importante ricostruire un modello di sicurezza in grado di allinearsi alle politiche adottate all'interno del mainframe [54];

- Oggi sono pochi i professionisti specializzati in sistemi legacy, mentre sono numerosi quelli con esperienza nella tecnologia cloud. Per colmare questo divario è necessario investire in formazione: da un lato fornire corsi sul mainframe per i dipendenti che lavorano con gli ambienti cloud, dall'altro aggiornare chi utilizza i mainframe. Ciò porta rischi, come ad esempio ritardi nei progetti di migrazione, errori di configurazione dovuti all'inesperienza e dipendenza da consulenti esterni [52].

  Tutte queste criticità mostrano come la migrazione dal mainframe al cloud non sia un processo banale e richieda un'attenta pianificazione.

4.3 Fasi della migrazione dei sistemi mainframe al cloud

Per affrontare queste complessità in modo efficace, è necessario pianificare con attenzione ogni passaggio della migrazione. Una roadmap ben strutturata consente di definire le priorità, valutare rischi e costi e stabilire i parametri di qualità attesi, garantendo così un percorso ordinato e consapevole. Le fasi di cui è composto un processo di migrazione sono principalmente sette, ognuna con obiettivi specifici [54].

4.3.1 Valutazione iniziale

E' la fase di partenza e serve ad avere una fotografia dettagliata dello stato in cui si trova l'ambiente mainframe. Innanzitutto, viene costruito un inventario con tutte le applicazioni attive, insieme ai moduli (blocchi di codice scritti in COBOL o PL/I) che li compongono, ognuno con le proprie funzioni. Si mappano le dipendenze sia a livello interno (tra i moduli) sia a livello esterno (tra le applicazioni). Dato che molto spesso la documentazione è scarsa, si ricorre a strumenti di analisi statica, in grado di eseguire attività di reverse engineering, ossia ricostruire le regole di business nascoste all'interno del codice. A questo si aggiunge la valutazione di altri componenti, cioè storage, rete, database legacy, e delle competenze del personale, per capire se sia necessario fornire maggiore formazione. Inoltre, si definiscono tutti i requisiti che la nuova soluzione cloud dovrà soddisfare, in termini di sicurezza, elasticità e performance. Infine, è necessario valutare attentamente i costi legati all'infrastruttura cloud, che dipendono dal reale utilizzo delle risorse [55]. Il risultato di questa fase è un report completo che mostra il grado di complessità del sistema e aiuta a prendere delle decisioni sensate nelle fasi successive [50].

4.3.2 Preparazione

Questa fase rappresenta il ponte tra l'analisi e l'avvio effettivo della migrazione. Innanzitutto, si vanno a definire i ruoli che deve ricoprire l'intero personale, creando un team di progetto specializzato sia in ambienti legacy che cloud, e in caso di gap di competenze vengono avviati dei programmi di formazione. Durante la migrazione i sistemi legacy potrebbero non essere disponibili, quindi per ridurre i disagi degli utenti si definisce una finestra temporale in cui i servizi legacy potrebbero non essere accessibili; tipicamente si sceglie una fascia temporale durante la notte o nel weekend cosicché l'interruzione sia quasi impercettibile [55]. Essendo la migrazione un processo non istantaneo, ci saranno dei momenti in cui una parte dei sistemi opera sul sistema legacy e la restante sul cloud, per questo si introduce un gateway di integrazione, ossia un sistema intermedio che riceve le richieste rivolte al legacy e le reindirizza verso il cloud o viceversa [55]. Viene anche definito un piano di rollback, in modo tale che se qualcosa va storto nell'ambiente cloud, si può tornare immediatamente indietro al vecchio sistema legacy [55]. Inoltre, vengono

scelti gli strumenti per la trasformazione del codice e per il monitoraggio, e in più il provider cloud a cui affidarsi. Si procede con una prima attività di sperimentazione: si migra soltanto una piccola applicazione non critica per verificare la compatibilità degli strumenti e stimare tempi e costi (Proof of Concept). Da questi risultati si riesce a capire che tipo di piano elaborare: quali applicazioni far migrare per prime e quali risorse allocare.

4.3.3 Definizione della strategia di migrazione

Una volta deciso cosa migrare e aver preparato il team, bisogna scegliere come migrare. Non tutte le applicazioni possono essere trattate nello stesso modo: bisogna scegliere l'approccio più adatto in base al livello di complessità e il grado di obsolescenza del codice di ogni applicazione. Vi sono 5 strategie possibili, che verranno approfondite nel paragrafo precedente:

- Rehosting: spostare senza modificare;
- Replatforming: spostare con adattamenti minimi;
- Refactoring: mantenere logica ma riscrivere il codice in linguaggi moderni;
- Rebuild: riscrittura da zero in architettura cloud-native;
- Replace: sostituire l'applicazione con una soluzione standard già esistente.

Tipicamente l'azienda sceglie un approccio ibrido, assegnando ad ogni applicazione l'approccio più adatto. In particolare, si procede con una migrazione ad ondate (Migration Waves): si parte dalle applicazioni meno critiche, e man mano che cresce la confidenza si passa ad applicazioni centrali.[50][52]

Per motivi di sicurezza alcune applicazioni rimangono nel sistema d'origine: in questa fase è importante stabilire quali applicazioni migrare e definire la loro distribuzione sui server cloud, tenendo conto di fattori come disponibilità e vincoli di rete [55].

Un aspetto cruciale è la scelta del fornitore cloud. Questo viene scelto tenendo conti di alcuni aspetti importanti:

- Costi complessivi dell'infrastruttura;
- Eventuali incompatibilità con il sistema legacy esistente;
- Rischio di vendor lock-in, ossia la dipendenza dal singolo fornitore;
- Disponibilità di servizi IaaS, PaaS e SaaS in grado di supportare architetture ibride;
- Tipi di framework e database effettivamente supportati;

- Posizione geografica dei data center e relativa latenza di rete;
- Garanzie in termini di sicurezza e conformità normativa.

In questo modo si riduce il rischio di scelte errate che, a lungo termine, potrebbero richiedere costosi adattamenti o un cambio di provider.

4.3.4 Preparazione e migrazione dei dati

Questa fase serve ad adattare il sistema legacy in modo che possa funzionare correttamente nel cloud, in linea con quanto deciso nella fase di progettazione. All'interno del mainframe i dati sono stati salvati in formati proprietari, quindi vengono convertiti in schemi compatibili con i database cloud. Si correggono eventuali discrepanze nel codice e si sviluppano integratori (wrapper), ossia componenti intermedi che consentono la comunicazione tra i sistemi legacy e il cloud senza modificare direttamente il codice esistente; in questo modo si adotta una soluzione meno invasiva [55]. E' importante che le applicazioni siano in grado di leggere e scrivere sugli stessi dati, quindi vengono condotti dei controlli rigorosi: viene fatto un confronto tra prima e dopo la migrazione contando i record, facendo delle verifiche a campione e dei test funzionali. Per rendere possibile ciò, è opportuno mantenere attivi entrambi i sistemi e, se necessario, predisporre un piano di roll back che consenta di evitare interruzioni operative. [52]

4.3.5 Test automatizzati

Una volta che i dati sono stati trasferiti, prima di procedere al passaggio effettivo al cloud è necessaria una fase di validazione, per garantire che le applicazioni si comportino come nel mainframe. Vengono, quindi, eseguite varie tipologie di test:

- Unitari: per verificare singoli moduli;
- Di regressione: per controllare che le nuove modifiche non abbiano introdotto errori;
- Di integrazione: per verificare che più componenti funzionino insieme nel modo corretto;
- Di performance: per testare tempi di risposta e scalabilità.

Inoltre, per rendere questo processo di migrazione più sicuro e affidabile, questi test vengono inseriti all'interno di pipeline di Continuous Integration e Continuos Delivery (CI/CD) che automatizzano il build, deployment e roll back, in modo da ridurre al minimo l'errore umano [52].

4.3.6 Cutover operativo & Monitoraggio

E' il momento del passaggio ufficiale al cloud, che può avvenire tramite un approccio "Big Bang", che è più rischioso poiché si sposta tutto rapidamente, oppure tramite un approccio più graduale, spostando le applicazioni in più fasi. A prescindere dalla strategia scelta, una volta che il sistema è attivo nel cloud, devono essere condotte della attività di monitoraggio continue: si controllano le performance, i costi e la sicurezza. Si raccolgono anche feedback da parte degli utenti, che sono in grado di individuare eventuali problemi di usabilità. [50][51]

4.3.7 Ottimizzazione continua

La migrazione non si conclude con la fase di Cutover, ma si trasforma in un processo di ottimizzazione. Il cloud, infatti, è un ambiente dinamico, in cui le risorse possono aumentare o diminuire in base alle esigenze degli utenti. Occorre ottimizzare le prestazioni, aggiornare i sistemi di sicurezza e introdurre nuove pratiche di governance. Solo in questo modo l'ambiente cloud può restare sicuro e performante nel lungo periodo.

4.4 Approcci di modernizzazione

Esistoni 5 possibili approcci di migrazione, introdotti da Gartner nel 2009 [56]:

- Rehosting (Lift and Shift): viene chiamato anche mainframe-as-a-service, e consiste nello spostare le applicazioni con modifiche minime in un ambiente nuovo, che può essere basato su cloud o un mainframe più recente e più flessibile gestito da un fornitore terzo. Si tratta di un processo rapido e a basso rischio, infatti permette di abbandonare subito l'hardware, mantenendo intatta la logica applicativa del mainframe, e impedisce l'interruzione delle attività perché non è necessario riscrivere tutto da zero. Lo svantaggio è che replicando l'ambiente originale ci si porta appresso tutti i limiti del vecchio sistema, perdendo così in agilità e in innovazione [52][53][54];

- Replatforming: consiste nello spostare le applicazioni mainframe in una piattaforma, con piccoli adattamenti, ad esempio cambiando database o middleware, senza riscriverle da zero. Si tratta di un approccio veloce e a basso costo, ma che lascia ancora diverse componenti legacy, limitando la piena agilità e capacità di innovazione [53][54];

- Refactoring: si mantiene la logica business legacy ma si ottimizza il vecchio codice per renderlo più adatto al cloud oppure si passa ad uno più moderno e più diffuso tra gli sviluppatori odierni (es. Java). In questo modo si raggiunge un compromesso tra la velocità dell'approccio "Lift and Shift" e la complessità della riscrittura totale.

Migliorano le prestazioni e la scalabilità, però allo stesso tempo è un approccio che richiede alti costi, poiché richiede del personale capace di conoscere sia le tecnologie mainframe che quelle cloud [52][53][55];

- Ricostruzione (Clean Slate): si riscrivono le applicazioni completamente da capo, in Java o su architetture cloud-native; quindi non si replica più il vecchio sistema ma si riprogetta, utilizzando linguaggi moderni. In questo modo si sfruttano al massimo i benefici del cloud, però si tratta di un approccio molto lungo e complesso, perché bisogna ricostruire tutte le logiche di business accumulate negli anni all'interno del mainframe [52][53][54];

- Sostituzione: si decide di non migrare affatto l'applicazione, ma di abbandonarla completamente e utilizzare direttamente un software moderno già pronto nel cloud, che svolge le stesse funzioni. Tutte le applicazioni presenti all'interno del mainframe vengono eliminate e sostituite da altre piattaforme. Potrebbe sembrare un approccio immediato, ma in realtà prima di metterlo in atto è necessario analizzare bene le dipendenze tra i vari moduli per non rischiare di bloccare i processi aziendali che dipendono da essi, e in più i flussi di lavoro esistenti devono essere ridisegnati [52][54].

Oltre alle 7 strategie originarie, AWS ha esteso il modello introducendone altri due:

- Retire: vengono dismesse tutte le applicazioni o i componenti che non offrono più valore. Ciò consente di ridurre i costi e semplificare il processo di migrazione [56];

- Retain: alcune applicazioni vengono mantenute nel sistema legacy perché troppo complesse per essere migrate, realizzando così la transizione delle applicazioni più urgenti. La migrazione delle applicazioni più critiche verrà però messa in atto in un secondo momento [56].

In particolare, nelle aziende viene spesso adottato un approccio ibrido, che non rappresenta una categoria distinta, ma bensì una combinazione di più approcci, scelti in base al livello di complessità delle applicazioni: quelle secondarie vengono migrate rapidamente tramite rehosting o replatforming, invece quelle più centrali vengono affrontate tramite refactoring o addirittura con una completa riscrittura [51][52].

A prescindere dall'approccio utilizzato, può risultare utile adottare il digital decoupling, che non rappresenta una strategia di migrazione autonoma ma una pratica complementare. In questo modello le applicazioni vengono spostate sul cloud (tramite riscrittura o adattamento), mentre i dati rimangono sul mainframe per motivi legati alla sicurezza o alla complessità del trasferimento. In questo modo l'azienda può introdurre innovazioni più rapidamente, senza rinunciare alla stabilità e all'affidabilità garantite dal mainframe [53].

Tra le tecniche di transizione dai sistemi legacy, esiste anche lo screen scraping, che è una soluzione utilizzata quando il mainframe non espone API o interfacce, rendendo difficile l'accesso ai dati interni. Consiste nel creare applicazioni che leggono automaticamente le informazioni a schermo e poi le riportano nel nuovo ambiente, simulando così le attività di un utente umano [56]. Nel contesto della migrazione verso il cloud, può essere impiegato per estrarre i dati dai sistemi legacy e riversarli in repository moderni, come ad esempio data lake o data warehouse cloud, consentendo di avviare il processo di transizione senza dover attendere lo sviluppo di interfacce più sofisticate. Lo screen scraping presenta però molti limiti: se cambia il layout l'automazione smette di funzionare, ed inoltre è difficilmente scalabile per grandi volumi di dati. Per queste ragioni viene considerata una soluzione temporanea, che in mancanza di alternative può ricoprire un ruolo immediato.

4.5 Esempi di casi studio

4.5.1 Dun & Bradstreet

Si tratta di una multinazionale leader nella fornitura di dati e analisi per il rischio aziendale e le decisioni commerciali. L'azienda ha basato per decenni la propria attività su sistemi mainframe e si è trovata di fronte alla necessità di modernizzare la propria infrastruttura per offrire servizi più rapidi ai propri clienti. Dopo un'attenta analisi, Dun & Bradstreet ha scelto di adottare un approccio di re-platforming, migrando le applicazioni e i carichi di lavoro verso Google Cloud Platform (GCP). Il progetto ha comportato la sostituzione delle soluzioni legacy con un'architettura cloud più flessibile e scalabile, capace di gestire query complesse e dataset di dimensioni elevate in tempi ridotti. L'obiettivo era duplice: ridurre i costi operativi derivanti dalla gestione dei sistemi legacy e allo stesso tempo incrementare la velocità e l'affidabilità delle elaborazioni. I risultati confermano la validità della scelta fatta:

- risparmio di 3,4 milioni di dollari nell'arco di cinque anni, con un ritorno sull'investimento (ROI) raggiunto in soli dieci mesi;
- riduzione del 96,3% dei tempi di elaborazione rispetto al mainframe;
- uptime del 99,992% con nessuna interruzione operativa. [57]

4.5.2 Consumer Good Company

Si tratta di un'azienda leader nel settore dei beni di largo consumo, operante da decenni con sistemi mainframe, che ha intrapreso un progetto di migrazione verso Amazon Web Services (AWS). La migrazione è stata eseguita tramite un approccio di rehosting e ha coinvolto 25 applicazioni mission-critical.
I benefici ottenuti sono i seguenti:

- Riduzione dei costi di supporto operativo del 25% e di quelli legati all'infrastruttura hardware del 65%;
- Rilascio delle funzionalità da una volta alla settimana col mainframe a più volte al giorno col cloud;
- Riduzione delle attività manuali nella fase di deployment dell'80% grazie all'introduzione delle pratiche DevOps.

Inoltre, la migrazione è stata completata con ben due mesi di anticipo, consentendo all'azienda di risparmiare ulteriormente. [58]

4.5.3 SSAB

Si tratta di una multinazionale dell'acciaio che fino a pochi anni fa basava le proprie attività su un'infrastruttura mainframe ormai datata. La strategia adottata non è stata una semplice sostituzione, ma un vero e proprio replatforming: il codice COBOL e i moduli legacy sono stati convertiti in C# .NET, i database IDMS migrati verso Azure SQL Server e le procedure batch tradizionali (JCL) sostituite da script automatizzati in PowerShell. Il nuovo ambiente è stato organizzato su una landing zone in Azure, con pipeline di integrazione e distribuzione continua (CI/CD) basate su Kubernetes e un approccio Agile per facilitare l'evoluzione e il mantenimento della piattaforma.
I risultati conseguiti hanno evidenziato un risparmio pari a circa 2,9 milioni di dollari in cinque anni, con un ritorno dell'investimento previsto in soli tre anni. Oltre ai benefici economici, l'azienda ha ottenuto un'infrastruttura più flessibile, con la possibilità di scalare rapidamente in funzione delle necessità di business. Inoltre, i team IT non sono stati sostituiti ma riqualificati sulle nuove tecnologie cloud. [59]

4.6 Dal mainframe al cloud nel progetto aziendale

Dopo aver analizzato alcuni casi di aziende che hanno affrontato la modernizzazione dei mainframe, in questa sezione viene descritto più nel dettaglio il progetto a cui sto prendendo parte. Come già evidenziato, la migrazione da sistemi mainframe a piattaforme moderne basate su architetture cloud rappresenta un processo complesso e strategico, poiché coinvolge non solo la dimensione tecnologica ma anche quella organizzativa, gestionale e di governance.

L'obiettivo del progetto è sostituire un sistema mainframe storico, dedicato alla gestione dei pezzi di ricambio, con una piattaforma cloud moderna, più scalabile, flessibile e integrata. Non si tratta quindi soltanto di abbandonare un'infrastruttura obsoleta e costosa, ma di rivedere l'intero modello operativo, modernizzare i processi, migliorare la qualità dei dati e ridurre i costi di manutenzione.

L'attuale sistema mainframe presenta numerose criticità: richiede costi di manutenzione sempre più elevati a causa delle spese hardware, utilizza logiche separate per la gestione dei processi legati ai dati e ai costi e necessita di continue integrazioni con altri sistemi aziendali, con conseguente aumento della complessità operativa. La nuova architettura in cloud mira a superare questi limiti attraverso una gestione più efficiente dei flussi informativi, la revisione e semplificazione delle logiche esistenti e una significativa riduzione delle interfacce da mantenere. Tutto ciò si traduce in maggiore efficienza, minori costi infrastrutturali e un miglior controllo sui dati e sui processi. Al momento ci troviamo soltanto nella fase iniziale di questa migrazione, dedicata principalmente alla valutazione dello stato attuale e alla pianificazione delle attività future.

4.6.1 Tipo di migrazione

L'approccio adottato si avvicinerà a un Rebuild, poiché i sistemi vengono ridisegnati e ricostruiti da zero con nuove logiche e architetture, senza limitarsi a un semplice spostamento del codice esistente. Per quanto riguarda la modalità di rilascio, non si tratta di un vero e proprio big bang né di una migrazione graduale e incrementale: il modello adottato è intermedio. Alcuni rilasci sono previsti in anticipo su specifiche componenti, mentre il go-live principale avverrà in un unico momento. Questo consente di ridurre i rischi di un passaggio unico e rapido, ma al tempo stesso necessiterà di tempi meno lunghi rispetto a una migrazione completamente frammentata.

4.6.2 Dati, processi e architettura

La migrazione prevede una revisione completa dei flussi informativi e dell'architettura sottostante. Non tutti i dati presenti sul sistema mainframe verranno trasferiti al nuovo ambiente: quelli obsoleti o non più necessari saranno esclusi.

Per comprendere meglio il processo è utile chiarire i principali attori coinvolti. Oltre al sistema mainframe e al futuro sistema cloud, è presente un sistema intermedio che ha un ruolo centrale nella gestione dei dati: questo rappresenta infatti il master delle informazioni sui pezzi di ricambio. Accanto a questi, vi sono poi diversi sistemi esterni, cioè applicazioni aziendali che utilizzano le informazioni aggiornate sui ricambi e sui costi per scopi operativi specifici, come la gestione degli ordini, dei magazzini, delle garanzie o delle analisi finanziarie.

Attualmente il flusso funziona così: il sistema intermedio invia i dati al mainframe, che li elabora, aggiunge informazioni supplementari (ad esempio codici e indicatori) e li distribuisce ai sistemi esterni. Il mainframe svolge quindi il ruolo di punto di snodo, raccogliendo, trasformando e smistando le informazioni.

Con l'introduzione del sistema cloud, questo flusso cambierà radicalmente: il sistema mainframe non manderà più le informazioni al sistema mainframe, ma alla nuova piattaforma cloud. In questo scenario, il cloud assumerà tutte le logiche e le funzionalità del vecchio sistema, aggiungendo strumenti più moderni come la tracciabilità delle modifiche, ossia registrerà chi ha effettuato una modifica, quando è stata fatta e quale cambiamento è stato applicato ai dati, così da avere sempre uno storico consultabile e una maggiore trasparenza e controllo. Diventerà quindi il nuovo punto di riferimento per i dati anagrafici ed esso stesso trasmetterà le informazioni ai sistemi esterni. I sistemi legacy continueranno a ricevere i dati con gli stessi formati utilizzati oggi, così da non interrompere i processi già in uso, mentre per i sistemi più moderni verranno create nuove interfacce più adatte al contesto cloud. Questo passaggio consentirà di eliminare la dipendenza dal mainframe, garantendo maggiore efficienza, scalabilità e integrazione con soluzioni tecnologiche moderne.

In particolare, i dati trasferiti dal sistema intermedio al sistema cloud saranno organizzati in file di tipo posizionale, ciascuno dedicato a uno specifico attributo oppure a un insieme di attributi correlati. Prima del caricamento nel cloud, tali file saranno sottoposti a processi di pulizia e validazione, con l'eventuale integrazione di attributi calcolati a partire da quelli già disponibili.

Una volta caricati, il team di progetto effettuerà verifiche puntuali per garantire coerenza e corretto allineamento. Per quanto riguarda le interfacce esterne, i layout attuali verso i sistemi legacy saranno mantenuti per garantire la continuità operativa, mentre per le integrazioni con sistemi moderni verranno sviluppati collegamenti ex novo.

La coerenza tra vecchio e nuovo sistema sarà assicurata da un approccio di "big bang controllato": il sistema mainframe verrà dismesso solo nel momento in cui il sistema cloud sarà piena mente operativo, riducendo così i rischi legati alla coesistenza di più ambienti paralleli.

4.6.3 Fasi e Tempistiche

Il piano di migrazione prevede diverse fasi scandite da mock test e attività di validazione:

- Primo mock test: previsto in anticipo e dedicato a una parte specifica dei dati e delle funzionalità, per verificare la capacità del sistema intermedio di garantire qualità e coerenza.
- Secondo mock test: più ampio e vicino al go-live ufficiale, servirà a validare l'intero set di dati e processi, verificando consistenza, completezza e corretto funzionamento delle interfacce.

4.6.4 Gestione della continuità operativa

Per garantire la continuità operativa durante la transizione, la migrazione prevede azioni mirate a ridurre al minimo i rischi. Da un lato, i dati presenti nel sistema mainframe verranno trasferiti al sistema intermedio e successivamente al sistema cloud, così da evitare perdite di informazioni. Parallelamente, verranno organizzate sessioni di revisione con i key users, per assicurare che tutti i processi critici vengano correttamente coperti dal nuovo ambiente. Infine, il progetto include attività di change management e formazione, con sessioni dedicate agli stakeholder aziendali, così da facilitare l'allineamento ai nuovi strumenti e favorire un'adozione graduale ed efficace.

4.6.5 Governance e qualità

Un aspetto centrale riguarda la governance dei dati e la gestione dei rischi. L'eventualità più critica identificata è la perdita di informazioni non note durante la migrazione. Per questo motivo, il sistema mainframe non verrà dismesso finché non sarà verificata la piena funzionalità del sistema cloud.

Gli eventuali disallineamenti verranno gestiti in due modi: manualmente, se dovuti a problemi di trasmissione, oppure tramite procedure automatizzate se riguardano errori di contenuto. Il sistema cloud integrerà regole di auditing per la tracciabilità delle modifiche e predisporrà piani di riallineamento.

Per quanto riguarda gli indicatori di successo (KPI), non sono ancora stati formalmente definiti, ma i controlli di coerenza e integrità sui dati costituiranno la metrica principale nelle prime fasi.

4.6.6 Change management

La migrazione non è soltanto un'attività tecnica, ma un progetto di trasformazione aziendale. In questo senso, il change management gioca un ruolo fondamentale. I key users saranno coinvolti fin dalle fasi di analisi e test, per validare dati e processi e ridurre i rischi di malfunzionamenti al momento del go-live.

La documentazione prodotta nel corso del progetto fungerà da base per la formazione e sarà affiancata da momenti di confronto con gli utenti, così da garantire un passaggio graduale ed efficace verso il nuovo sistema cloud.

4.6.7 Sintesi

In conclusione, la migrazione dal mainframe al cloud nel progetto aziendale non rappresenta soltanto un'operazione tecnica, ma un vero e proprio percorso strategico che ridisegna

processi, architetture e responsabilità. Grazie a un approccio rebuild, al ruolo centrale del sistema intermedio, alla pianificazione di mock test mirati e a un rilascio gestito in più fasi, sarà possibile garantire la qualità delle informazioni, ridurre i rischi e guidare l'organizzazione verso un ambiente cloud più moderno, scalabile e integrato.

Capitolo 5

Azure Data Factory

Nel progetto di migrazione da mainframe a cloud, un ruolo centrale è stato affidato ad Azure Data Factory, il servizio di integrazione dati di Microsoft. Si tratta di uno strumento intuitivo e visivo che permette di creare e orchestrare pipeline per spostare e trasformare grandi volumi di dati provenienti da sistemi legacy. In particolare, è stato utilizzato per gestire il flusso di dati dal sistema intermedio al nuovo sistema cloud, costituendo quindi lo strumento chiave per questo obiettivo.

ADF è stato scelto perché consente di gestire in maniera ordinata l'intero flusso: dall'acquisizione dei dati dal sistema intermedio, alla loro validazione, fino all'aggiornamento delle nuove tabelle nel cloud. Oltre alla semplicità d'uso, offre funzionalità avanzate come l'integrazione con Key Vault per la sicurezza, GitHub per il versionamento e CI/CD per l'automazione dei rilasci, garantendo così affidabilità e scalabilità.

In questo modo Azure Data Factory è diventato il motore operativo della migrazione, capace di trasformare e propagare i dati assicurandone coerenza e tracciabilità lungo tutti i processi.

5.1 Componenti principali di Azure Data Factory

Azure Data Factory è composto da diversi elementi che, combinati tra loro, consentono di progettare e gestire in modo completo i flussi di dati.

5.1.1 Pipeline

L'elemento chiave di Azure Data Factory è la pipeline, ossia un contenitore logico al cui interno vengono definite una serie di attività concatenate o parallele, necessarie per spostare e trasformare dati in maniera automatizzata [62].

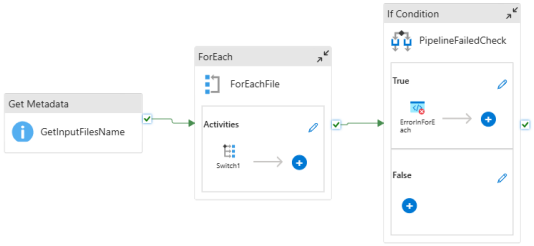

Figura 5.1. Esempio di pipeline su Azure Data Factory

Le attività possono essere suddivise in tre macrocategorie:

- Data movement activities: si occupa di spostare i dati da una fonte ad una destinazione, senza modificarne il contenuto. L'attività principale è "Copy Data", che legge un dataset di input (source) e lo copia in un dataset di output (sink) tramite uno schema di mapping delle colonne con possibile conversione automatica del tipo di dato. Può gestire il caricamento massivo di grandi dataset e supporta vari tipi di formati (es. CSV, Parquet, JSON, ecc.), la loro compressione e ricodifica [62].

- Data transformation activities: ha la funzione di modificare i dati per renderli utilizzabili. Una delle attività appartenenti a questa macrocategoria è "Mapping Data Flow", che consente di effettuare trasformazioni complesse tramite interfaccia visiva, quindi senza l'utilizzo di codice. Le operazioni messe a disposizione sono: unione dataset, aggregazioni, filtraggi di record, creazione di nuove colonne, ecc. [63]

- Control flow activities: non lavorano direttamente sui dati ma servono a gestire l'orchestrazione del flusso di una pipeline, cioè stabiliscono come e in che ordine eseguire le varie operazioni. Le principali attività sono:

  – ForEach: permette di iterare su un insieme di elementi, cioè il processamento di più file o tabelle uno per volta;

  – IfCondition: permette di eseguire altre operazioni in base ad una condizione booleana;

  – Switch: si utilizza quando ci sono più percorsi di esecuzione e si vuole scegliere quello corretto in base al valore di un parametro;

  – Until: è simile al ciclo While, cioè ripete una serie di attività finché una condizione è vera;

  – Web Activity: permette di chiamare API esterne;

  – Lookup: lavora su un dataset aggiornandolo o recuperandone un valore;

  - Get Metadata: ottiene metadati da file, cartelle o tabelle (es. nome, dimensione, ecc.) prima del caricamento;
  - Set Variable: imposta variabili interne;
  - Append Variable: aggiunge un valore ad una variabile di tipo array;
  - Execute Pipeline: permette di richiamare un'altra pipeline all'interno di quella principale. [62]

5.1.2 Dataset

Per costruire le nostre pipeline all'interno di Azure Data Factory bisogna definire dei dataset, ossia oggetti logici che descrivono i dati da utilizzare: la loro posizione, il loro formato e le modalità di accesso.

All'interno delle nostre pipeline sono state utilizzate principalmente due tipologie di dataset:

- File grezzi in formato testo, salvati all'interno dei container, ospitati nello spazio di archiviazione di Azure, ossia Azure Blob Storage, che rappresentano l'area di atterraggio per i dati che provengono da fonti esterne. [63]
- Tabelle relazionali PostgreSQL contenute all'interno di un database PostgreSQL, che rappresenta lo spazio di archiviazione per i dati strutturati. Non si può accedere a queste tabelle direttamente su Azure ma tramite uno strumento esterno ad Azure nel quale è possibile eseguire delle query, nel nostro caso DBeaver. [64]

Per poter accedere ai dati è necessario definire i Linked Service, che rappresentano delle porte d'ingresso verso i sistemi da cui i dati vengono letti. Quando viene avviata una pipeline e si vuole attingere a dei file o delle tabelle, senza un Linked Service non si saprebbe dove trovarli. Nel caso dei file grezzi il Linked Service permette di collegarci al Blob Storage, mentre nel caso dei dati strutturati consente l'accesso al server PostgreSQL. I Linked Services contengono tutte le informazioni necessarie alla connessione, come parametri di configurazione e credenziali di autenticazione. [64]

5.1.3 Trigger

I trigger sono meccanismi che avviano in maniera automatica l'esecuzione di pipeline in determinati istanti di tempo o in base all'accadimento di un evento. [62]

- Scheduled Trigger: attiva le pipeline ad orari prestabiliti (es. ogni ora, giorno, settimana o giorno del mese) ed è pensata per scenari semplici di automazione, però se un'esecuzione viene persa non viene recuperata. [63]

- Tumbling Window Trigger: attiva la pipeline in finestre fisse (ad esempio ogni ora o ogni giorno). Se un'esecuzione viene persa può essere recuperata in seguito. È ideale per scenari di elaborazione incrementale. [63]
- Event-based Trigger: le pipeline si attivano in risposta ad eventi, solitamente legati allo storage, come l'arrivo di un file in un container. È utile per scenari in cui i dati devono essere elaborati non appena disponibili. [63]

5.1.4 Integration Runtime

In Azure Data Factory, l'Integration Runtime rappresenta un motore di calcolo che permette l'esecuzione delle attività nelle pipeline e in più rende possibile il collegamento alle diverse origini o destinazioni dati [60]. Esistono tre principali tipologie di IR, scelte in base allo scenario applicativo:

- Azure Integration Runtime viene completamente gestito da Microsoft e consente di spostare dati e di eseguire trasformazioni direttamente nel cloud;
- Self-hosted Integration Runtime (SHIR) è invece installato su macchine virtuali all'interno di ambienti aziendali protetti, e si utilizza quando non si vuole che i dati siano esposti a internet;
- Azure-SSIS Integration Runtime permette infine di eseguire pacchetti SSIS già esistenti su infrastrutture Azure, e viene adottato quando esistono già processi ETL sviluppati in SSIS e si vogliono migrare al cloud senza doverli riscrivere da zero. [60][61]

5.1.5 Monitoraggio e gestione delle esecuzioni

Durante l'esecuzione delle pipeline, è possibile monitorarne costantemente lo stato in tempo reale accedendo alla sezione Monitor, segnalando eventuali errori [60][62].

E' possibile scendere nel dettaglio, analizzando il risultato di ogni attività contenuta all'interno della pipeline. In caso di errore, è possibile avviare nuovamente la pipeline correggendo tutti gli errori segnalati.

In particolare, ad ogni run viene registrato il successo/fallimento, la data e l'ora dell'avvio e la durata dell'esecuzione. Inoltre, vengano raccolti tutti gli indicatori utili per ottimizzare i flussi, ossia quantità di dati letti e scritti, la velocità di throughput, il tempo di elaborazione. Tutti questi dati vengono raccolti in un contenitore (Log Analyitcs Workspace) dove è possibile eseguire query avanzate per individuare cause di errori, o anche creare regole di alert, ad esempio se una pipeline fallisce più di 3 volte consecutive, viene mandata una notifica in automatico [60][63].

Il monitoraggio è fondamentale nei processi di migrazione perché tenendo traccia di tutte le esecuzioni mantiene sotto controllo le prestazioni dei flussi ETL. Inoltre, il sistema per la diagnostica permette di evitare di controllare manualmente ogni pipeline del portale e di reagire immediatamente a problemi grazie alle notifiche. [60]

5.1.6 Sicurezza e Key Vault

In Azure Data Factory la sicurezza viene gestita tramite Azure Key Vault, un servizio che archivia e protegge password, chiavi di accesso, certificati. Un altro pilastro è l'RBAC (Role-Based Access Control) che autorizza l'accesso delle risorse solo alle persone autorizzate. [60]

5.1.7 CI/CD e integrazione con Git

Azure Data Factory si integra con GitHub, consentendo di salvare le modifiche di pipeline, dataset o servizi non direttamente nell'ambiente di Data Factory, ma in un repository Git. In questo modo ogni cambiamento è tracciato e reversibile: è sempre possibile tornare a una versione precedente in caso di errore. Inoltre, più sviluppatori possono lavorare contemporaneamente sugli stessi flussi senza rischiare di sovrascriversi a vicenda, infatti ogni sviluppatore lavora sul proprio branch. Quando il codice relativo ad ogni branch viene considerato stabile le modifiche effettuate da ogni singolo sviluppatore vengono unite nel branch principale. [60]

Questa integrazione con Git è alla base del processo di CI/CD (Continuous Integration/Continuous Deployment):

- Con la Continuous Integration (CI), ogni modifica viene salvata e gestita nel repository Git, garantendo controllo delle versioni e collaborazione tra sviluppatori.
- Con la Continuous Deployment (CD), le pipeline validate vengono distribuite automaticamente nei vari ambienti di Azure Data Factory, cioè prima in test e poi in produzione. In questo modo si evita di dover ricopiare manualmente le pipeline da un ambiente all'altro, riducendo così errori e garantendo coerenza tra gli ambienti.

5.1.8 Mapping Data Flow

All'interno di Azure Data Factory, il Mapping Data Flow rappresenta un ambiente visivo che ci permette di effettuare delle trasformazioni sui dati. In particolare, consente di costruire processi ETL (Extract, Transform, Load) complessi senza la necessità di dover scrivere l'intero codice: tramite un'interfaccia grafica è possibile definire in modo intuitivo le regole di trasformazione. [60]

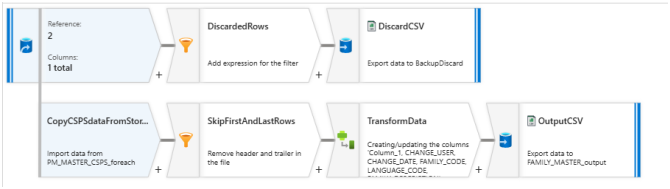

Figura 5.2. Esempio di Mapping Data Flow

Dal punto di vista architetturale, i Data Flow si basano su Apache Spark, che rappresenta il motore di calcolo su cui vengono eseguite le varie trasformazioni. Ciò consente di elaborare grandi volumi di dati in parallelo, garantendo prestazioni elevate e scalabilità automatica.

Tra le principali operazioni che possono essere realizzate all'interno del Data Flow troviamo:

- Filtri e selezione dei dati, per mantenere solo le righe che soddisfano determinate condizioni;
- Join tra dataset provenienti da origini diverse;
- Aggregazioni e calcoli statistici;
- Derivazione di nuove colonne, tramite funzioni predefinite;
- Pivot e unpivot per riorganizzare la struttura dei dati;
- Sorting e ordinamenti personalizzati;
- Lookup per arricchire i dati con informazioni aggiuntive da altre tabelle;
- Sink multipli, cioè la possibilità di scrivere i dati trasformati su destinazioni diverse all'interno dello stesso flusso.

L'aspetto interessante è che l'utente può disegnare graficamente tutte queste operazioni in sequenza, tramite blocchi collegati tra di loro, mentre il motore sottostante genera automaticamente il codice Spark necessario. In questo modo si unisce la semplicità di un approccio low-code con la potenza di un'elaborazione distribuita.

Ad esempio, consideriamo un file contenente le vendite giornaliere in formato CSV. Con un Mapping Data Flow è possibile:

1. Leggere il file da uno storage;
2. Eliminare le righe non valide (ad esempio quelle con valori nulli);

3. Unirle con un dataset contenente l'anagrafica dei prodotti;
4. Calcolare per ogni prodotto il totale delle vendite settimanali;
5. Salvare il risultato in una tabella SQL o in un data lake.

Il Mapping Data Flow diventa quindi uno strumento perfetto quando si devono gestire trasformazioni complesse in modo intuitivo, riducendo la necessità di codice personalizzato e garantendo la scalabilità offerta dal cloud.

5.2 Architettura e implementazione pipeline

Nel progetto di Migrazione abbiamo adottato come strumento Azure Data Factory per implementare le pipeline che automatizzano il trasferimento dei dati. In particolare ci siamo occupati del flusso di dati dal sistema intermedio al nuovo sistema cloud.

I file di ingresso sono suddivisi per flusso, ciascuno con un contenuto, frequenza di invio e perimetro dati propri. In base al flusso di appartenenza, ogni file viene indirizzato verso pipeline dedicate, che seguono sempre lo stesso ordine: fase di ingestion, processamento e invio al sistema di destinazione. Le pipeline presentano una struttura simile, con alcune variazioni legate al flusso di appartenenza. In particolare, distinguiamo due tipi di flussi:

- Indipendenti (master): gestiscono le anagrafiche "autonome" (es. pezzo, modello, ecc.) ed il loro contenuto non richiede che altri flussi siano stati già caricati per essere valido;
- Dipendenti (associazioni): descrivono relazioni che fanno riferimento ad una o più anagrafiche (es. associazione modello-pezzo), quindi il loro contenuto richiede l'esistenza dei corrispondenti record master per poter essere valido.

I file sono di tipo posizionale e arrivano a frequenze diverse in base al tipo di flusso a cui appartengono. Si tratta di file di testo a larghezza fissa, in cui i campi occupano un intervallo di caratteri fisso, e non esistono delimitatori tra di essi. Sono caratterizzati da una riga iniziale e finale (header e trailer) contenenti meta-info, cioè data, conteggio righe, ecc. Sono file molto rigidi, poiché se cambia la larghezza di un campo cambia tutto il layout, e anche difficilmente leggibili a occhio dato che non ci sono dei delimitatori che separano i vari campi, di conseguenza è necessario ristrutturarli.

I file possono arrivare in due modi distinti:

- Full: rappresentano un'istantanea completa e aggiornata del set di dati disponibile al momento dell'invio. Ogni nuovo file sostituisce integralmente il precedente, includendo tutti i record correnti e riflettendo eventuali aggiunte, modifiche o eliminazioni

rispetto agli invii precedenti. E' più pesante ma più conveniente poiché se viene perso durante la consegna il suo contenuto può essere facilmente recuperato all'invio del successivo;

- Delta: contengono soltanto le variazioni rispetto all'invio precedente, ossia i record di nuova creazione, aggiornati o eliminati. E' più leggero ma se viene perso durante la consegna non può essere più recuperato.

La frequenza con cui arrivano i file dipende da un insieme di fattori tecnici:

- Volatilità dei dati: vengono inviati più frequentemente i dati che cambiano spesso e che quindi servono freschi a valle; invece se cambiano raramente la frequenza può essere abbassata;
- Tipo di file: i file in full sono pesanti quindi vengono inviati con una minor frequenza, mentre i file in delta, essendo più leggeri, possono girare spesso;
- Dipendenze: se un flusso dipende da un altro si fa in modo che i due flussi dipendenti non abbiano troppa differenza di frequenza, per evitare di riempire il recycle e generare rielaborazioni inutili.

5.2.1 Ingestion

La prima fase è quella di ingestion e ha lo scopo di acquisire i file dal sistema intermedio ed effettuare dei controlli.

Il file che arriva dal sistema intermedio è un file di tipo posizionale, che viene ristrutturato tramite due possibili modalità:

- Mapping Data Flow (Figura 5.3): questa attività è ideale per file grandi e a bassa frequenza e racchiude al suo interno una serie di passaggi a cui viene sottoposto il file di partenza:
  1. Il file posizionale viene letto come un'unica colonna;
  2. Vengono scartate la riga iniziale e finale tramite un filtro sulla lunghezza;
  3. In ogni record vengono estratti i vari campi tramite la funzione substring, in base alle posizioni definite a priori;
  4. Il file finale viene inserito all'interno di un container di output.
- Doppio Copy Data (Figura 5.4): questo approccio viene adottato per file piccoli e ad alta frequenza ed è composto da due attività:
  1. Nel primo Copy Data il file di partenza viene copiato e caricato all'interno di una tabella temporanea con una sola colonna contenente gli interi record;

2. Nel secondo Copy Data su tale tabella viene eseguita una query che estrae i campi sempre attraverso la funzione substring e filtra il trailer e l'header.

Una volta ristrutturato, il file viene copiato all'interno di una tabella temporanea, utilizzata per salvare il file grezzo che arriva in quel momento. Tutti i campi sono di tipo *varchar* e non sono definite chiavi (né primarie né esterne); in questo modo è possibile salvare tutti i record, anche quelli con errori. Questa tabella non è utilizzata per altri scopi e viene svuotata ad ogni esecuzione.

La tabella temporanea, ossia il file strutturato inviato dal sistema intermedio, va sottoposta ad una serie di controlli di qualità. Le righe con errori bloccanti (non contengono la chiave, hanno uno o più campi obbligatori mancanti o contengono dei campi con formati non validi) verranno eliminati e successivamente salvati all'interno di una tabella degli scarti, insieme al codice errore, motivazione, timestamp, così da garantirne la tracciabilità. Esistono anche gli errori non bloccanti, ossia righe parzialmente recuperabili (es. manca un attributo che verrà gestito successivamente da un'altra pipeline). In questi casi la riga non viene eliminata ma marcata con un warning e messa in recycle, per essere processata al run successivo.

Una volta fatti tutti i controlli, sia per i flussi dipendenti che per quelli indipendenti si salvano in una tabella di trigger tutte le modifiche introdotte del nuovo file: per ogni record (chiave) che viene inserito, aggiornato o eliminato, viene scritto il tipo di evento nella tabella tramite il campo *change type*. In questo modo si dispone di un punto unico da cui leggere e analizzare i cambiamenti, senza dover confrontare nuovamente l'intera tabella master con il file ricevuto. In particolare se il file è in full bisogna eseguire una query che confronta le due tabelle, individuando quelle aggiunte, cancellate e cambiate; mentre se è in delta basta copiare la tabella cosi com'è perche contiene già il file con l'attributo *change type*.

Una volta terminate queste operazioni di controllo e tracciabilità, si recupera il file di partenza che quindi viene inserito all'interno di un container di backup e contemporaneamente eliminato dal container di partenza che dovrà ospitare un nuovo file al prossimo run.

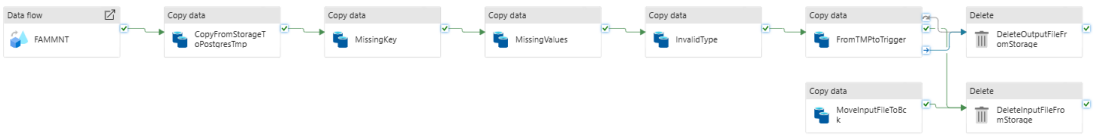

Figura 5.3. Pipeline di ingestion con Mapping Data Flow

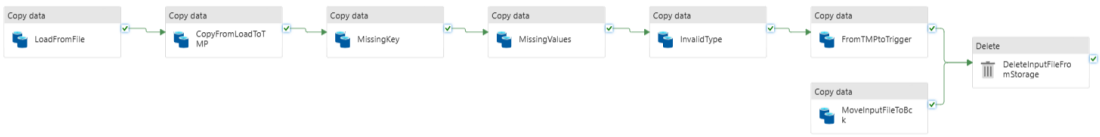

Figura 5.4. Pipeline di ingestion con doppio Copy Data

5.2.2 Processamento

Dopo l'ingestion e la validazione, i dati entrano nella fase di processamento, in cui le variazioni vengono applicate alle tabelle master e, se necessario, propagate verso altri flussi.

Il processamento non è uguale per tutti i flussi, ma varia a seconda della sua tipologia. Alcuni sono indipendenti e quindi si limitano ad aggiornare direttamente la propria tabella master. Altri sono master e hanno il compito di propagare modifiche che possono influenzare più domini ad essi collegati. Infine, ci sono i flussi di associazione, che collegano entità diverse e devono gestire con attenzione le dipendenze tra loro.

1. I flussi indipendenti sono i più semplici da elaborare. La tabella di trigger, popolata in fase di ingestion, contiene già tutte le variazioni da applicare. Il processamento quindi si limita a:

   - selezionare le righe con variazioni effettive (inserimenti e aggiornamenti);
   - aggiornare direttamente la tabella master corrispondente.

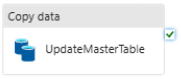

Figura 5.5. Pipeline di processamento per un flusso indipendente

   Non essendoci dipendenze, non è necessario propagare ulteriori informazioni o alimentare altre tabelle trigger. Ad esempio, immaginiamo un flusso che gestisce il dizionario dei codici tecnici. Se arriva un nuovo codice, oppure un codice già esistente cambia descrizione, questa variazione viene registrata nella tabella trigger e inserita direttamente nella tabella master. Non servono ulteriori passaggi, perché i codici tecnici non hanno effetti a cascata su altri flussi.

2. I flussi master sono più complessi, perché costituiscono la base dati principale da cui dipendono tutti gli altri flussi. Qui la tabella trigger non viene utilizzata solo per

aggiornare la tabella master, ma anche per alimentare altre tabelle trigger in modo da propagare le modifiche anche ai flussi correlati e per attivare calcoli derivati.

Figura 5.6. Pipeline di processamento per un flusso master

Per capire meglio, forniamo i seguenti esempi:

- Cancellazione di un modello: quando un modello viene eliminato, non basta cancellarlo dalla tabella master dei modelli, ma occorre provvedere anche all'eliminazione di tutti i pezzi di ricambio ad essi associati. Per questo motivo, l'informazione della cancellazione viene scritta in una tabella trigger del flusso corrispondente all'associazione pezzo-modello. In questo modo si evita che restino pezzi collegati a un modello inesistente. Quando verrà avviata la pipeline relativa al flusso pezzo-modello nella trigger ci sarà già l'informazione sul pezzo-modello eliminati;
- Aggiornamento delle date di produzione: se cambia la data di fine produzione di un modello, questa informazione deve riflettersi anche sui pezzi a esso collegati. L'evento viene quindi scritto in una tabella trigger a livello di pezzo, che sarà poi utilizzata in un flusso successivo per ricalcolare le date di produzione dei pezzi interessati. In questo modo il sistema si limita ad aggiornare solo i dati realmente coinvolti, evitando rielaborazioni massicce e inutili;
- Ricalcolo del flag di regione attiva: ogni modello è associato a una o più regioni in cui è commercializzato o in cui il servizio è attivo. Quando la regione di un modello cambia, questa informazione non rimane circoscritta al solo modello, ma deve essere propagata anche ai pezzi di ricambio collegati. Per questo motivo, la modifica viene intercettata e scritta in una tabella trigger dedicata. Successivamente, il sistema avvia un processo di ricalcolo che aggiorna il flag di regione attiva per tutti i pezzi associati a quel modello. In questo modo si assicura che i dati siano coerenti: un pezzo risulta disponibile solo nelle stesse regioni in cui il modello collegato è effettivamente attivo.

3. I flussi di associazione mettono in relazione domini diversi, come ad esempio i modelli con i pezzi di ricambio o i modelli con le linee di prodotto. La loro complessità risiede nel fatto che non possono aggiornare le relazioni senza verificare prima che i campi associati esistano già nelle rispettive tabelle master. Infatti il processamento viene diviso in due fasi:

(a) Per ogni record si controlla che i campi collegati siano già stati caricati nelle loro rispettive tabelle master;

(b) Se anche solo uno dei campi non è ancora disponibile, il record non viene scartato, ma rimane in tabella con uno flag che lo contrassegna come "non processabile". In questo modo potrà essere ripreso automaticamente al successivo run, quando i dati mancanti saranno stati caricati (Recycle).

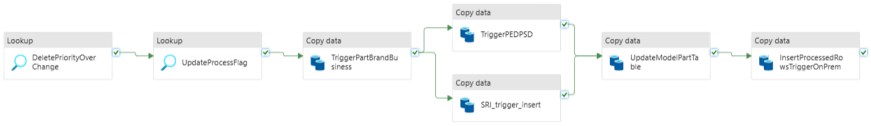

Figura 5.7. Pipeline di processamento per un flusso di associazione

Forniamo i seguenti esempi:

- Associazione modello-pezzo: supponiamo che arrivi un record che collega un nuovo modello a un pezzo. Se sia il pezzo che il modello sono già presenti rispettivamente nella tabella master dei pezzi e dei modelli, l'associazione può essere creata subito. Se invece il pezzo o il modello non sono ancora stati caricati, l'associazione rimane in attesa. Al successivo run, non appena le tabelle master dei pezzi e dei modelli saranno aggiornate, la relazione verrà processata senza dover reinviare il file;
- Associazione modello-linea di prodotto: quando viene inserito un nuovo legame tra un modello e una linea di prodotto, non ci si limita a registrarlo nella tabella delle associazioni. Poiché questa variazione influisce sul calcolo dei codici di brand e di business dei pezzi collegati, l'evento viene propagato in una tabella trigger dedicata ai ricalcoli. In questo modo il sistema non ricalcola tutto, ma solo i pezzi effettivamente coinvolti dalla nuova relazione.

5.2.3 Controlli comuni

A prescindere dal tipo di flusso, ci sono regole trasversali che guidano il processamento:

- Uso delle tabelle trigger: fungono da buffer intermedio fra file ricevuto e tabella master, permettendo di gestire le modifiche in modo ordinato e di propagare le informazioni senza dover confrontare nuovamente intere basi dati;
- Gestione dei conflitti: se nello stesso ciclo si verificano più eventi sulla stessa entità, vengono applicate regole di priorità (ad esempio, una cancellazione prevale sempre su una modifica);

- Ordine e idempotenza: le operazioni seguono sempre lo stesso ordine (inserimenti, aggiornamenti, cancellazioni) e sono progettate per non introdurre inconsistenze anche in caso di run ripetuti;
- Audit e tracciabilità: ogni variazione viene registrata con timestamp, utente e tipo di cambiamento, consentendo riconciliazioni e analisi a posteriori;
- Pubblicazione esterna: quando previsto, le righe già processate vengono inviate a sistemi esterni tramite apposite tabelle di uscita, mantenendo distinti il processamento interno e la distribuzione esterna.

Grazie a questa architettura, ogni cambiamento è gestito in modo ordinato, tracciabile e sicuro, assicurando che il sistema rimanga coerente e robusto anche in presenza di scenari complessi.

Capitolo 6

Architetture dei Dati: Data Warehouse, Data Lake e Data Lakehouse

I dati hanno sempre ricoperto un ruolo essenziale all'interno di molte organizzazioni, poiché consentono di individuare quali sono le tendenze di mercato e le preferenze dei clienti [67]. Ciò porta a prendere decisioni sensate, che aiutano le aziende a trarre vantaggio competitivo rispetto alla concorrenza. In particolare, un sistema di gestione dei dati deve esser in grado di unificare, validare e distribuire in maniera efficiente un'enorme quantità di dati. Tutto ciò è finalizzato al data mining, ossia all'estrazione di informazioni utili [67].

Negli ultimi decenni abbiamo assistito ad un'evoluzione tecnologica senza precedenti, che ha portato ad una crescente produzione di dati, di ogni tipo e proveniente da una moltitudine di fonti: i cosiddetti "big data". Ciò ha spinto molte aziende a sviluppare sistemi di gestione dei dati e di business intelligence sempre più efficienti, dove l'elaborazione, l'archiviazione, la gestione e l'analisi dei dati sono diventati sempre più veloci e semplici [67]. Questa innovazione tecnologica ha portato ad una democratizzazione: questi nuovi sistemi possono essere utilizzati da chiunque, non più solo da utenti qualificati ma anche da quelli meno esperti [65].

In base allo scenario applicativo esistono diversi tipi di sistemi di gestione e analisi dei big data: uno dei più rilevanti è il data lakehouse, un approccio adottato all'interno del progetto aziendale a cui sto prendendo parte [67].

Si tratta di una piattaforma unificata, nata per combinare i punti di forza di altri due sistemi di gestione dei dati: la flessibilità di archiviazione del data lake con le capacità di

analisi del data warehouse.

Per comprendere pienamente il sistema di data lakehouse è utile prima approfondire il data warehouse e il data lake.

6.1 Data Warehouse

Il data warehouse è stato introdotto alla fine degli anni '80 dai ricercatori IBM Barry Devlin e Paul Murphy per il supporto alle decisioni [67].

Si tratta di sistemi progettati per archiviare grandi volumi di dati, provenienti da diverse fonti, utilizzando uno schema predefinito (schema on-write) [65]. Questa struttura chiara e organizzata dei dati rende questo sistema ideale per supportare attività di business intelligence all'interno di ogni contesto aziendale.

Tradizionalmente, un data warehouse era ospitato on-premise, su un computer mainframe; invece oggi numerosi data warehouse si basano su una tecnologia cloud. Nel primo caso l'azienda deve comprare e installare i server fisici (hardware, storage, rete), invece nel secondo questa prima fase di configurazione spetta alla piattaforma cloud di cui si serve il business [67].

Alcuni dei settori che hanno tratto vantaggio dai progressi del data warehousing sono [67]:

- Settore manufatturiero: gestione delle spedizioni e servizio clienti;
- Settore finanziario: gestione dei sinistri, valutazione dei rischi, analisi della fatturazione;
- Settore logistico: gestione delle flotte dei veicoli;
- Settore delle telecomunicazioni: dati sulle chiamate;
- Settore sanitario: analisi dei dati clinici.

6.1.1 Architettura del data warehouse

Di seguito viene illustrato il funzionamento di un sistema di datawarehouse nella sua interezza, per offrire una visione completa del flusso di dati.

Lo schema in Figura 6.1 mostra il flusso dei dati dalle fonti all'ultimo livello del processo, in cui i dati sono pronti per interfacciarsi con l'utente finale.

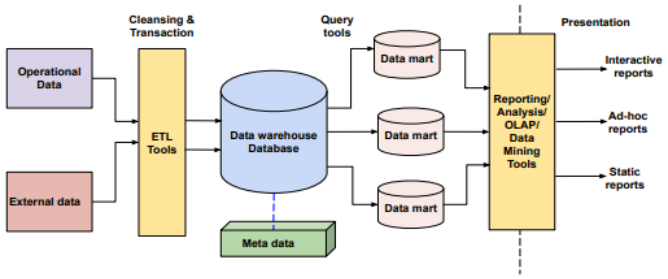

Figura 6.1. Flusso dei dati dalle fonti al front-end in un Data Warehouse [67]

1. Le fonti da cui si attinge prima che i dati vengano ospitati nel data warehouse sono:

   - sistemi operativi interni all'azienda (es. ERP, CRM, gestionali) che usano database transazionali in cui i dati sono già organizzati in forma tabellare [68];
   - dati grezzi provenienti da fonti esterne (es. file di log, file CSV, dati di sensori).

   In particolare, i dati grezzi vengono memorizzati in posizioni specifiche del disco sottoforma di file chiuso, ossia non visibile all'utente: avendo una struttura interna proprietaria del motore di database possono essere letti solo dal database stesso. In base al tipo di dato salvato i file possono essere strutturati (es. tabelle con righe e colonne), semi strutturati (es. JSON, XML) e strutturati in modo approssimativo (es. immagini, log, audio). Inoltre, possono essere a riga (row-based) dove ogni riga contiene tutti i valori di un record, oppure a colonna (column-based), dove ogni colonna è salvata separatamente. In particolare, il formato colonnare è più vantaggioso in termini di prestazioni, infatti se ad esempio si vuole fare un'operazione su un unico dato basta leggere solo la colonna che lo contiene e non l'intero record [65].

   Tutti i file precedentemente salvati vengono legati logicamente tra di loro in un'unica tabella ordinata, con righe e colonne, che aggiunge dei metadati. A differenza del formato file visibile solo internamente al sistema, il formato tabellare può essere letto dagli utenti e rende possibile l'esecuzione di query usando tecniche come il partizionamento (dividere i dati in blocchi più piccoli) e clustering (ordinare i dati in base ad una colonna) [65].

2. I dati nel formato tabellare possono essere sottoposti ad un processo ETL (Extract, Transform, Load) [65]:

   - Extract: i dati vengono estratti dalle fonti sottoforma di tabelle dai database transazionali o sotto forma di file che però vengono subito convertiti in forma tabellare;

- Transform: i dati vengono puliti, unificati e arricchiti all'interno di tabelle temporanee in cui vengono rimosse le righe con determinate caratteristiche (chiavi duplicate, dati non validi, campi obbligatori mancanti) e vengono fatti dei calcoli su alcuni campi;
- Load: i dati trasformati vengono caricati nel sistema di data warehouse, sottoforma di tabelle pronte per l'analisi.

Questo approccio è un metodo tradizionale, utilizzato principalmente nei sistemi di data warehouse on-premise, dove si vuole già validare i dati prima che arrivino al sistema di destinazione.

Un altro approccio possibile è ELT (Extract, Load, Transform) che viene utilizzato nei sistemi di data warehouse moderni basati su cloud [65]. A differenza del metodo precedente, una volta che i dati sono stati estratti dalle fonti e convertiti in formato tabellare, vengono direttamente caricati all'interno del data warehouse, dove si sfrutta la potenza del motore cloud per effettuare le trasformazioni. Questo approccio è molto più flessibile rispetto al precedente in quanto è possibile archiviare qualsiasi tipo di dato prima di trasformarlo.

Inoltre, grazie all'architettura MMP (Massively Multi-Processing), che caratterizza i data warehouse basati sul cloud, le trasformazioni possono essere eseguite velocemente all'interno del sistema poiché il carico di lavoro viene distribuito in più nodi [65]. Utilizzando questo approccio si evita di eseguire trasformazioni inutili a monte, ma solo quelle che effettivamente servono. Nei sistemi di datawarehouse tradizionali, invece, veniva applicata un'architettura SMP (Symmetric Multi Processing) che eseguiva il calcolo in una macchina sola, costituita da più processori [65].

3. Al termine del processo ETL o ELT, i dati vengono memorizzati nel data warehouse, che funge da repository centralizzato e strutturato, progettato per supportare analisi avanzate come Business Intelligence, data mining e intelligenza artificiale.

   All'interno del data warehouse agiscono due componenti principali: il motore di archiviazione e il motore di calcolo.

   Il motore di archiviazione regola il meccanismo interno dei dati, cioè gestisce le operazioni CRUD [65]:

   - Create: scrivere i dati all'interno dei file;
   - Read: leggere i dati quando vengono eseguite delle query da parte degli utenti;
   - Update: aggiornare i dati quando vengono apportate modifiche;
   - Delete: eliminare dati ormai non più necessari.

Più nel dettaglio, decide in quale posizione del disco memorizzare i dati nei file, mantiene aggiornate le strutture dei dati (indici, vincoli di integrità come *non null* e chiavi esterne). Inoltre, utilizza meccanismi di archiviazione efficienti, tra cui compressione e ottimizzazione dell'accesso ai dati [65].

Il motore di calcolo, invece, si occupa di prelevare i dati memorizzati dal motore di archiviazione ed eseguire le query richieste dall'utente, in particolare gestisce tutte le attività computazionali finalizzate a generare informazioni utili per le organizzazioni:

- aggregazioni (es. calcolo della media, somme, raggruppamenti per data o area geografica);
- trasformazione dei dati (es. applicazione di filtri, join tra le tabelle, ordinamenti).

Utilizza tecniche volte alla massimizzazione delle prestazioni e alla riduzione dei tempi di esecuzione delle query [65].

Infine, per aiutare gli utenti ad orientarsi tra i dati, viene costruito un catalogo di metadati, che include i nomi delle tabelle, schema, origine, caratteristiche dei dati, ecc., facilitando così il processo di ricerca da parte degli utenti. Però si tratta di uno strumento difficile da scegliere poiché non esiste uno standard chiaro [67].

4. Il passo successivo riguarda la creazione dei data mart, ossia un sottoinsieme di dati del data warehouse più mirato che contiene solo i dati necessari per un determinato settore. Ad esempio, se un data warehouse contiene i dati relativi ad una banca intera, il data mart ha solo i dati di una filiale. A volte può essere del tutto indipendente dal data warehouse ed essere direttamente costruito a partire dalle fonti [67].

   Dataset più piccoli migliorano le prestazioni delle query, rendendo l'analisi più veloce, evitando che l'utente debba visitare l'intero data warehouse. Tuttavia, l'estrazione di un data mart a partire da un data warehouse può essere costosa e complessa, poiché servono processi ETL aggiuntivi ed inoltre è necessario sincronizzare i dati tra i due sistemi, aumentando così il carico di gestione [67].

5. I data warehouse offrono un' interfaccia front-end intuitiva con cui l'utente comunica per estrarre insight. Questa interazione è resa possibile da API che permettono all'utente di collegarsi al data warehouse e fare richieste per ottenere determinati dati. Gli strumenti di cui dispone l'utente in questo livello del data warehouse sono [67]:

   - Strumenti di query: consentono agli utenti del business di interrogare il data warehouse, senza dover conoscere a fondo la struttura del database e il linguaggio SQL;

- Strumenti di reportistica: aiutano gli utenti a costruire report regolari, utili per monitorare le attività del business;
- Strumenti di data mining: facilitano la ricerca di relazioni, non evidenti ad occhio nudo o tramite l'uso di query, all'interno di grandi set di dati;
- Strumenti OLAP: sfruttano il concetto di data multidimensionale e permettono l'analisi dei dati secondo più dimensioni. In particolare, ne esistono di due tipologie:
  - MOLAP (OLAP Multidimensionale): i dati sono pre-aggregati e organizzati all'interno di un cubo multidimensionale, quindi quando arriva una richiesta da parte di un utente i risultati sono immediatamente disponibili [68];
  - ROLAP (OLAP Relazionale): le query SQL vengono generate dinamicamente per rispondere alle richieste dell'utente [68].

6.1.2 Funzionalità del datawarehouse

- Governance e sicurezza: si tratta di aspetti fondamentali per garantire il corretto utilizzo delle informazioni archiviate da parte degli utenti. Trattandosi di un repository centrale che raccoglie dati a partire da una moltitudine di fonti, il data warehouse deve disporre di strategie efficienti volte a proteggere i dati. In generale un sistema di gestione di dati qualsiasi deve adattarsi a determinate normative. Ad esempio, GDPR è una legge dell'Unione Europea che regola il trattamento dei dati personali dei residenti nell'Unione Europea e si applica a tutte le organizzazioni che trattano dati personali di cittadini europei, indipendentemente dalla loro posizione geografica [72]. Per garantire la conformità al GDPR:
  - Vengono applicate tecniche di crittografia sia ai dati a riposo che a quelli in transito, per proteggere l'accesso da parte di utenti non autorizzati [65][72];
  - Si controlla se un determinato utente abbia il permesso di accedere ad un determinato contenuto, attraverso meccanismi come il controllo degli accessi basato sui ruoli. È possibile nascondere alcune righe o colonne a determinati utenti, ad esempio un utente può individuare solo record relativi alla propria area geografica [65];
  - Si registrano tutte le attività effettuate sui dati tramite audit log, per tenere traccia di chi ha apportato modifiche ai dati e quando [72];
  - Si tracciano l'origine e il percorso dei dati, facilitando la gestione dei consensi e la risposta alle richieste degli utenti (data lineage) [72].

  Poiché all'interno del data warehouse i dati sono strutturati e centralizzati, è più facile applicare controlli di accesso, auditing e garantire la sicurezza dei dati.

Bisogna anche sottolineare che man mano che i dati all'interno del sistema aumentano le organizzazioni potrebbero dover aggiornare i principi di governance dei dati e le misure di sicurezza.

- Gestione della concorrenza: sullo stesso set di dati possono essere eseguite contemporaneamente più azioni, cioè può accadere che più utenti vi accedano mentre è già in esecuzione un processo di caricamento, trasformazione o interrogazione sugli stessi dati. Il sistema di data warehouse è in grado di gestire questa situazione, disponendo di efficienti capacità di gestione dei carichi di lavoro che riallocano il carico su diverse macchine che in parallelo svolgono le varie attività, evitando così di violare il livello di servizio aziendale (SLA) [65].

- Gestione delle transazioni: durante le operazioni sui dati effettuate da più utenti, è necessario verificare che non vengano commessi errori che possano compromettere le analisi sui dati finali. Nel data warehouse ciò viene gestito efficacemente, poiché le transazioni rispettano le proprietà ACID [65]:

  – Atomicità: ogni transazione deve essere trattata come una singola unità, cioè deve essere eseguita nella sua interezza, altrimenti viene annullata in modo da non introdurre incongruenze;

  – Coerenza: le modifiche apportate dopo una transazione devono essere eseguite all'interno di tutto il sistema;

  – Isolamento: se più transazioni vengono eseguite in parallelo non devono interferire tra di loro;

  – Durabilità: le modifiche apportate devono essere mantenute nel tempo.

  In questo contesto è utile definire degli strumenti che aiutano a mantenere queste proprietà:

  – Rollback: se una pipeline che ha lo scopo di caricare i dati da un sistema sorgente fallisce a metà è necessario annullarla completamente tornando allo stato precedente;

  – Locking: se più utenti vogliono accedere allo stesso tipo di dato si esegue un blocco in modo tale che non vi siano conflitti tra gli utenti [65].

- Latenza delle query: il tempo che intercorre tra una richiesta da parte dell'utente e la risposta del sistema è una questione da non sottovalutare in un contesto aziendale che vuole ricavare in maniera rapida informazioni dai dati al fine di eseguire delle analisi. In questo contesto risultano essenziali le seguenti tecniche:

  – Indicizzazione: invece di scorrere lungo tutto il data warehouse l'utente consulta l'indice che lo porta subito al punto giusto;

  - Partizionamento: si divide una tabella in più parti, in base ad un criterio che può essere, ad esempio, una data o un'area geografica;
  - Meccanismi di caching: i risultati delle query vengono temporaneamente salvati, così in caso di richiesta degli stessi dati non è necessario eseguire di nuovo lo stesso calcolo.

Tutte queste tecniche hanno lo scopo di facilitare i processi di ricerca da parte degli utenti e quindi ridurre al minimo i tempi di risposta da parte del sistema di data warehouse [65].

L'evoluzione all'interno di un contesto aziendale e quindi anche dei requisiti rende necessario che il sistema di data warehouse si mostri flessibile e in grado di adattarsi ai cambiamenti. Le modifiche dello schema possono riguardare l'aggiunta di un nuovo campo e quindi di una nuova colonna oppure la modifica di una colonna già presente all'interno di una tabella. Tutto ciò non impatta sulle prestazioni ed è reso possibile all'interno del data warehouse, a patto che le modifiche attuate siano conformi alle versioni precedenti, in modo da rispettare la struttura rigida stabilita a priori [65].

6.1.3 Limiti del data warehouse

Nonostante i data warehouse offrano efficienti capacità analitiche e rappresentino ottimi strumenti di business intelligence, in grado di rispondere a query sofisticate nel minor tempo possibile, presentano comunque una serie di limiti da non sottovalutare.

- Gestione inefficiente dei Big Data: negli ultimi anni il progresso tecnologico ha portato alla generazione di grandissime quantità di dati. I data warehouse non sono in grado di gestire in maniera efficiente volumi dell'ordine di terabyte o addirittura petabyte. Infatti, importare milioni o miliardi di dati può rallentare il sistema e portare ad una diminuzione delle prestazioni, o anche una semplice query eseguita su grandi quantità di dati può diventare un'operazione lenta e onerosa in termini di risorse.
- Limitazione nell'uso con il Machine Learning: il data warehouse è stato progettato per archiviare dati in forma strutturata, dove vengono trasformati, aggregati e organizzati in tabelle con colonne e tipi di dato ben definiti, in modo da alimentare report, grafici e dashboard. Questo aspetto li rende particolarmente adatti alla business intelligence, ma poco indicati a tecniche di Machine Learning, oggi sempre più diffuse:
  - Il Machine Learning lavora anche (e soprattutto) con dati semi strutturati (come JSON o XML) e non strutturati (testo libero, immagini, video, audio), che il data warehouse non gestisce in modo efficiente;

  - Il Machine Learning spesso lavora con dati grezzi, che non sono disponibili nel data warehouse, dove i dati sono già pre-processati per analisi tradizionali;
  - L'addestramento dei modelli richiede calcoli complessi e numerose iterazioni su grandi volumi di dati. Il data warehouse è invece pensato per letture rapide, non per elaborazioni pesanti;
  - Forzare il data warehouse ad affrontare questi carichi comporta limitazioni di scalabilità e un aumento significativo dei costi, soprattutto in ambienti non cloud-native.

- Scalabilità rigida nei sistemi tradizionali: nei data warehouse tradizionali on-premise il sistema di archiviazione e di calcolo vengono gestiti all'interno della stessa infrastruttura, quindi se è richiesta maggiore potenza di calcolo, dovuta all'esecuzione di query complesse, è necessario aumentare sia le risorse di calcolo che quelle di storage, in quanto strettamente collegate. Questo problema è stato però superato grazie all'introduzione di sistemi di data warehouse basati su cloud (es. Snowflake, BigQuery o Redshift) in cui lo storage e il calcolo sono due sistemi indipendenti e ciò rende possibile una scalabilità elastica, aumentando così le prestazioni e riducendo i costi [65].

- Vendor lock-in: nei data warehouse i dati vengono memorizzati in formati proprietari, ossia utilizzabili solo all'interno del sistema stesso. Questo fenomeno si chiama "vendor lock-in" e costringe l'azienda ad utilizzare solo gli strumenti del proprio fornitore in quanto esportare i dati diventa tecnicamente limitante [65].

- Costi di archiviazione e gestione: nonostante i sistemi di data warehouse cloud prevedano costi ridotti, questi aumentano man mano che i volumi di dati crescono e che si effettuano query complesse. Quando vengono create tabelle pre-aggregate e viste materializzate che velocizzano le analisi, viene occupato più spazio e ciò porta ad un aumento dei costi di archiviazione. I costi legati all'archiviazione fanno sì che molte aziende memorizzino solo alcuni set di dati e ciò porta ad analisi poco accurate. Altri costi da non sottovalutare sono quelli legati alla gestione delle pipeline di ETL/ELT [65].

- Assenza dei dati in tempo reale: oggi molte aziende richiedono dati aggiornati in tempo reale per dashboard, decisioni operative, ecc. Nei data warehouse è praticamente impossibile che le informazioni visibili siano quelle in tempo reale poiché prima di essere memorizzati all'interno del sistema i dati devono essere sottoposti a un processo di ETL/ELT. Questi processi non girano continuamente, ma tipicamente ogni notte o ogni ora, di conseguenza quando i dati vengono interrogati possono essere vecchi di ore o giorni (dati obsoleti) [71].

6.2 Data Lake

Tutti i limiti del data warehouse hanno spinto la maggior parte delle organizzazioni a generare soluzioni più flessibili per la gestione dei dati. In particolare, negli ultimi anni grazie ai progressi tecnologici, abbiamo assistito ad una crescente quantità dei dati a nostra disposizione e soprattutto ad un aumento dei vari tipi di dati (dati di sensori, transazioni commerciali basati sul web, social media ecc.). In risposta a questi fenomeni si è presentata la necessità di sviluppare una serie di sistemi in grado di archiviare qualsiasi tipo di dato, sia strutturato che non strutturato. Tra questi, uno dei più rilevanti è il data lake.

Si tratta di un ambiente di archiviazione centralizzato, che ospita dati grezzi, nel loro formato originale, ossia:

- dati strutturati (es. relazionali, fogli excel, tabelle SQL);
- dati semi-strutturati (es. JSON, XML, file log);
- dati non strutturati (documenti di testo, immagini, video, file audio, e-mail);
- dati binari (es. file eseguibili, immagini in formato raw, file compressi);
- dati di streaming (es. dati di sensori, log di accesso in tempo reale).

A differenza dei data warehouse, nei data lake non viene definito uno schema ben preciso prima che i dati vengano memorizzati all'interno del sistema, ma viene applicato solo dopo l'archiviazione, quando i dati vengono utilizzati (schema on-read). Non è quindi più prevista una trasformazione preliminare. Questa non necessità di predisporre a priori un'attenta pianificazione della struttura porta ad una notevole riduzione dei costi [67].

I data lake, disponendo di uno storage economico, possono memorizzare anche dati storici o poco utilizzati. Inoltre, sono scalabili poiché le risorse di storage e quelle di calcolo sono due entità separate, che lavorano una indipendentemente dall'altra.

Una caratteristica importante dei data lake è l'archiviazione di file in formato aperto: i dati possono essere analizzati o utilizzati da qualsiasi strumento, indipendentemente dal software utilizzato. Esempi di formati aperti sono Apache Parquet, ORC, Avro, ecc.

6.2.1 Architettura del Data Lake

Nonostante si tratti di un unico repository centralizzato, il data lake può essere suddiviso in diversi livelli per meglio comprenderne il suo funzionamento, mostrati in Figura 6.2 [67]:

- Livello dei dati grezzi: i dati non elaborati arrivano nel modo più rapido ed efficiente possibile, senza essere sottoposti ad alcun tipo di modifica;

- Livello dei dati standardizzati: si tratta di un livello opzionale, utilizzato solo in alcune implementazioni, dove i dati vengono convertiti nel formato a loro più adatto e serve a migliorare le prestazioni nel trasferimento dei dati dal livello dei dati grezzi a quello pulito;
- Livello dei dati puliti: i dati vengono trasformati in set utilizzabili per l'analisi, cioè vengono puliti, denormalizzati e organizzati per scopo, tipo e struttura del file;
- Livello applicativo: è il livello di produzione a cui possono accedere gli utenti finali, dove viene applicata la logica di business e dal quale si attinge per gli strumenti di business intelligence e per le tecniche di machine learning.

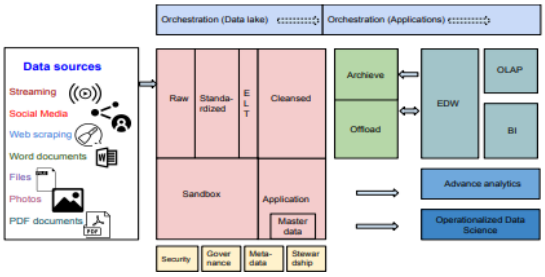

Figura 6.2. Flusso dei dati nel Data Lake: dalle fonti alla preparazione per l'analisi [67]

6.2.2 Caratteristiche del Data Lake

Il data lake può essere definita un'architettura dinamica, non solo perché è in grado di memorizzare tanti tipi di dati diversi, ma anche perché è in grado di adattarsi alle esigenze aziendali in continua evoluzione (nuovi tipi di dati, nuovi strumenti, nuove regole di sicurezza, ecc.). Inoltre, può funzionare su diverse piattaforme, sia quelle cloud che quelle on-premise (nei server fisici dell'azienda). Infine, può collegarsi a diversi strumenti di Machine Learning per effettuare analisi predittive.

I dati, contenuti nel data lake, essendo grezzi non hanno alcun tipo di significato, di conseguenza viene costruito un catalogo di metadati, ossia informazioni sui dati, come ad esempio significato semantico dei dati, formati e proprietà dei dati e politiche sui dati, che aiutano gli utenti ad avere una comprensione più ampia dei dati. Questi rivestono un ruolo cruciale più nei data lake, in quanto i dati presenti non hanno una struttura chiara e ordinata come nel caso del data warehouse. In questo contesto si inseriscono figure come i data steward che si occupano di mantenere aggiornati i metadati, ma anche assicurarsi che i dati siano corretti e coerenti [67].

All'interno dei data lake sono archiviati anche dati master, cioè informazioni di riferimento su entità presenti all'interno del sistema. Vengono utilizzati durante i processi di analisi o trasformazione dei dati, ad esempio se si sta analizzando l'ordine di un cliente può essere utile avere tutte le informazioni relative al cliente [67].

I data lake, se integrati con strumenti di streaming analytics offrono la possibilità di analizzare dati in tempo reale, poiché i dati grezzi vengono memorizzati immediatamente, dopo l'estrazione.

In molti casi i data lake possono supportare i data warehouse, contribuendo ad una gestione più efficiente dei dati [68]:

- I data lake possono conservare dati storici che provengono dai sistemi di data warehouse, in questo modo alleggeriscono il loro storage, migliorando così le prestazioni e riducendo i costi di archiviazione all'interno del data warehouse;

- I data lake possono contribuire a gestire carichi di lavoro pesanti eseguiti all'interno dei data warehouse. Ad esempio, i processi ETL possono diventare molto onerosi e richiedere una grande potenza computazionale, quindi parte di questo carico di lavoro viene spostato nei data lake, e una volta terminato il processo di pulizia i dati vengono restituiti al data warehouse, pronti per essere analizzati.

6.2.3 Casi d'Uso del Data Lake

Di seguito vengono illustrati i principali scenari applicativi in cui i data lake possono operare.

I data lake sono particolarmente adatti per attività di Machine learning, che hanno bisogno di grandi volumi di dati non strutturati e soprattutto non filtrati, come testo, immagini, dati di sensori per addestrare modelli robusti [67].

Inoltre, possono essere utilizzati per sperimentare nuove tecnologie prima che vengano messe in pratica in un progetto reale (Proof of Concept), risparmiando così tempo e risorse. Questo perché non è richiesta una struttura iniziale rigida e quindi possono essere utilizzati liberamente durante l'esplorazione. Una volta che l'esperimento restituisce risultati validi, i dati possono essere migrati nel data warehouse [68].

Infine, il data lake può fungere da archivio di dati grezzi, nel loro formato nativo (data backup). Ciò può essere sfruttato quando si verificano guasti hardware e si vogliono recuperare i dati persi, oppure quando il business vuole verificare il lavoro effettuato da team precedenti [68].

6.2.4 Limiti del Data Lake

I data lake presentano diversi limiti, che è importante conoscere per comprenderne le criticità nella gestione e nell'analisi dei dati. Di seguito illustriamo i principali.

I data lake non possono essere utilizzati direttamente per supportare strumenti di Business Intelligence poiché devono prima essere preparati tramite un processo ETL o ELT [68].

Se molte aziende riescono a estrarre valore aziendale dai propri dati, altre, invece, se non sono in grado di applicare adeguate pratiche di qualità dei dati, rischiano di imbattersi in fenomeni come il data swamp. Il data swamp (palude di dati) si verifica quando i metadati non vengono aggiornati in maniera costante, oppure quando i dati non vengono sottoposti a controlli periodici o anche quando le trasformazioni ritardano [68].

Una delle conseguenze che porta ad un data swamp all'interno di un data lake è il non supporto alle transazioni ACID. Infatti, si possono presentare i seguenti casi all'interno di un data lake:

- Un'operazione che vuole modificare un determinato dato su più file si interrompe a metà, in questo modo alcuni file risultano aggiornati e altri no;
- Se più processi scrivono su uno stesso dataset non esiste nessun meccanismo che impedisca l'accesso concorrente;
- Se un'operazione fallisce, non esiste un meccanismo che riporta tutto allo stato precedente.

Trattandosi di sistemi progettati per accettare qualsiasi tipo di dato, la sicurezza e la governance sono aspetti particolarmente critici, soprattutto nella fase iniziale dell'architettura. In particolare, gestire dati non strutturati o semi-strutturati richiede attenzione particolare per garantire la protezione dei dati personali e la conformità al GDPR. Devono essere definite delle regole di governance per non rischiare di incorrere in fenomeni come il data swamp, quindi i dati devono essere etichettati tramite utilizzo di metadati per favorirne la tracciabilità. Inoltre, devono essere definite tecniche di crittografia per proteggere tutti i dati sensibili e controlli degli accessi per fare in modo che le normative su privacy e protezione dei dati vengano rispettate [67].

I data lake si contraddistinguono per le loro elevate capacità di archiviazione ma inevitabilmente se i volumi di dati crescono in modo significativo le prestazioni possono peggiorare. Ad esempio, dataset enormi aumentano i tempi di risposta a query poco complesse oppure i processi di ricerca da parte degli utenti possono rallentare in presenza di metadati troppo ingombranti.

I data lake, pur offrendo grande flessibilità nella memorizzazione di dati, presentano limitazioni in alcune applicazioni avanzate come il Natural Language Processing, l'elaborazione audio, la visione artificiale e, più in generale, nell'addestramento di modelli di deep learning su dataset non tabulari. Questi limiti derivano dal fatto che i data lake conservano i dati nel loro formato nativo, senza classificazione automatica. Ne consegue una latenza elevata nei processi di lettura ed elaborazione e la mancanza di strumenti nativi per la gestione semantica dei dati. Per superare queste limitazioni è necessario integrare il data lake con sistemi di catalogazione, gestione dei metadati, strumenti di orchestrazione e motori esterni di machine learning in grado di trasformare i dati grezzi in informazioni utilizzabili.

6.3 Differenze principali tra data warehouse e data lake

Di seguito vengono riassunte le principali differenze tra data warehouse e data lake.

1. Un data warehouse archivia solo dati strutturati mentre un data lake memorizza dati sia strutturati che non strutturati, nel loro formato nativo. In particolare, nel primo caso lo schema con cui vengono organizzati i dati è definito a priori (schema on-write), mentre nel secondo viene applicato solo quando i dati vengono utilizzati, dopo l'archiviazione (schema on-read).

2. I file all'interno di un data warehouse vengono archiviati in formati chiusi, cioè proprietari, leggibili principalmente con il software del vendor. Nel data lake, invece, i file sono memorizzati in formati aperti, cioè utilizzabili da qualsiasi strumento, senza vincoli legati al software specifico.

3. I costi che hanno a che fare con l'estrazione e la successiva archiviazione del data warehouse sono elevati poiché è richiesta una definizione preliminare dello schema ed inoltre deve essere eseguito un processo ETL prima dell'archiviazione. Al contrario, i costi di archiviazione del data lake sono ridotti poiché i dati vengono ospitati nel loro formato nativo.

4. I data warehouse possono essere utilizzati da chiunque, sia da personale qualificato che inesperto, invece i data lake, ospitando dati non ancora elaborati, richiedono personale con competenze appropriate.

5. I data lake memorizzano i dati grezzi, che non essendo organizzati in strutture relazionali non sono facilmente interrogabili dagli utenti; invece nei data warehouse il formato tabellare consente agli utenti di leggere i dati in maniera chiara ed eseguire query usando linguaggi SQL.

6. Quando viene costruito un data warehouse è importante che il business abbia già chiaro quello che è lo scopo del sistema, per sprecare meno storage possibile, dati i costi elevati. Al contrario, quando viene generato un data lake ci si può permettere di non sapere con precisione l'obiettivo del business, quindi è possibile memorizzare all'interno del sistema dati inizialmente inutili, che però potrebbero servire in un secondo momento, oppure dati storici che verranno utilizzati per analisi successive.

7. I data warehouse, grazie alla loro struttura in formato tabellare, offrono efficienti capacità di analisi dei dati, quindi sono strumenti ideali per la business intelligence; invece i data lake, memorizzando dati grezzi, si prestano per costruire modelli predittivi tramite tecniche di machine learning.

8. I data warehouse lavorano con dati che vengono archiviati dopo aver subito un processo ETL. Per questa ragione non permettono l'utilizzo di dati in tempo reale: l'analisi avviene con un certo ritardo rispetto a quando sono stati estratti dalle fonti. Al contrario, i dati all'interno del data lake vengono salvati immediatamente quindi, se integrati con opportuni strumenti, permettono l'analisi in tempo reale.

6.4 Data Lakehouse

Ormai vengono prodotti sempre più dati di diverso tipo, dimensione e velocità e ci si è resi conto che i modelli tradizionali non sono in grado di gestire efficacemente questa complessità. Nei paragrafi precedenti abbiamo visto, infatti, che sia data warehouse che data lake presentano alcuni limiti importanti: i data warehouse non gestiscono dati non strutturati, prevedono alti costi e non supportano tecniche di machine learning; i data lake non dispongono di tecniche di governance e sicurezza adeguate che portano inevitabilmente a veri e propri "data swamp", e non sono ottimizzati per supportare attività di business intelligence.

Per superare questi limiti, spesso le aziende utilizzano una combinazione di più sistemi diversi: un data lake per archiviare dati grezzi, un data warehouse per le analisi e altri sistemi specializzati per streaming, immagini, grafici, ecc. Questo approccio, però, porta inevitabilmente ad un aumento dei costi e ritardi nell'esecuzione delle query, poiché è necessario copiare i dati tra i vari sistemi per garantire coerenza [69]. Inoltre, ai dati potrebbe essere applicato uno schema diverso in ogni sistema e devono essere gestiti processi ETL/ELT su più sistemi, aumentando così la probabilità di bug e guasti [71].

Per poter affrontare al meglio questa situazione, Databricks ha sviluppato un nuovo sistema di gestione dei dati: il Data Lakehouse [69]. Questo nuova piattaforma combina i punti di forza di entrambi i modelli: lo storage economico e flessibile del data lake e la

qualità, governance e velocità del data warehouse. In questo modo, le aziende possono gestire i propri dati all'interno di un unico sistema, senza duplicazioni.

6.4.1 Architettura del Data Lakehouse

Un data lakehouse è fisicamente costituito da un data lake: tipicamente uno storage scalabile che può ospitare dati strutturati e non strutturati. Sopra lo storage vengono aggiunte tutte le funzionalità tipiche del data warehouse.

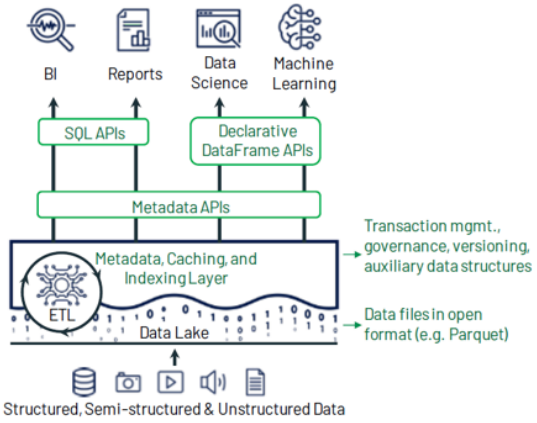

Figura 6.3. Possibile implemetazione di un Data Lakehouse [73]

Prima di descrivere nel dettaglio le caratteristiche del data lakehouse illustriamo il processo a cui sono sottoposti i dati in un sistema di data lakehouse [70].

1. Ingestione: i dati strutturati, semi strutturati e non strutturati arrivano da varie fonti (database transazionali, applicazioni CRM, database, ecc.). Spesso si usa un processo ETL/ELT per pulire i dati prima di salvarli.

2. Archiviazione: i dati grezzi vengono salvati sotto forma di formato di file aperto e standardizzato, proprio come avviene in un data lake, ad esempio Apache Parquet o JSON. Sono leggibili e utilizzabili da una grande varietà di strumenti, sia per la business intelligence che per applicazioni di Machine Learning, senza la necessità di convertire il formato passando da un sistema all'altro [69]. Inoltre, l'uso di formati aperti e compressi consente di ridurre al minimo i costi di storage.

3. Livello di metadati: il data lake da solo è semplicemente un archivio di oggetti dove anche una semplice operazione, come un aggiornamento su più file, non è atomica. Di conseguenza sopra il data lake viene aggiunto un livello di metadati che conferisce ai file una forma tabellare che mantiene il formato aperto. In questo modo il data lake si trasforma quasi in un database proprio come il data warehouse, diventando

facilmente interrogabile. In particolare i metadati sono informazioni sui dati, ovvero lo schema, i file che appartengono a quella tabella, nome delle colonne, i tipi di dati che contengono. Grazie al livello dei metadati è possibile avere pieno controllo delle operazioni sui file, infatti:

- accettano solo valori che rispettano lo schema;
- assicurano che le transazioni eseguite sui dati siano conformi alle proprietà ACID;
- durante l'accesso ai dati controllano che l'utente abbia l'autorizzazione, e tiene traccia di chi ha letto, modificato o caricato i dati (audit);
- creano delle copie virtuali dei dati senza duplicarli fisicamente (cloning a copia zero);
- tengono traccia delle versioni dei dati, in modo da poter interrogare i dati di una certa data (time travel);
- salvano temporaneamente i dati più utilizzati in un cache, dove vengono transcodificati in un formato ordinato e veloce da leggere dalla CPU, quando deve eseguire delle query [69];
- contengono dati ausiliari, come indici, per trovare rapidamente i dati rilevanti senza leggere tutti i file, e statistiche (minimi, massimi, conteggi ecc.) [69][71].

4. Accesso alle query: le API si interfacciano con il livello dei metadati e sono il mezzo tramite il quale l'utente esegue query sui dati. L'uso di diversi tipi di API consente di soddisfare esigenze diverse:

   - SQL APIs: consentono di interrogare i dati con il linguaggio SQL;
   - Declarative DataFrame APIs: ricevono le richieste in modo dichiarativo dagli utenti e costruiscono un piano di esecuzione ottimizzato, per poi eseguire per davvero le operazioni [71].

5. **Casi d'uso**
   I dati possono essere usati dagli utenti finali per:

   - BI (Business Intelligence): dashboard e analisi operative;
   - Reports: reportistica periodica;
   - Data Science: analisi avanzata dei dati;
   - Machine Learning: modelli predittivi e algoritmi di AI.

6.4.2 Caratteristiche del data lakehouse

1. All'interno del data lakehouse è possibile interrogare direttamente i dati grezzi senza la necessità di dovergli conferire preventivamente una struttura. Anche se esistono casi in cui per assicurarsi che le analisi siano di qualità sia necessario sottoporre i dati a processi ETL/ELT, che però risultano meno complessi rispetto a quelli utilizzati nel data warehouse [71]. Ciò garantisce tempi di risposta più brevi.

2. Il data lakehouse è un repository centralizzato che contiene tutti i dati originali, eliminando la necessità di copie di dati tra sistemi diversi. Questo approccio riduce notevolmente i costi di storage, semplifica il controllo sugli accessi e consente agli utenti di accedere ai dati più recenti senza la necessità di nessuna trasformazione o sincronizzazione tra sistemi diversi.

3. Il data lakehouse consente a più utenti di interrogare i dati contemporaneamente, adattando dinamicamente le risorse in base al carico di lavoro.

4. Nei sistemi basati sui file è necessario riscrivere l'intero file per modificare lo schema, invece nel formato tabella nel data lakehouse la modifica del sistema viene gestito dai metadati, eliminando così la necessità di ricomporre l'intera tabella.

5. In un data lake tradizionale più pipeline di dati che leggono e scrivono contemporaneamente nello stesso set di dati possono generare delle incongruenze nei dati. Il data lakehouse, invece, supporta le transazioni ACID, proprio come il data warehouse [69].

6. I data lakehouse, proprio come i data warehouse, dispongono di tecniche di governance e sicurezza adeguate, che sono in grado mantenere la qualità dei dati alta. Ad esempio, si controllano gli accessi degli utenti in base al ruolo che svolgono all'interno dell'azienda, e vengono anche registrate le modifiche effettuate dagli utenti tramite tabelle di audit.

7. Il data lakehouse, ospitando sia dati grezzi che strutturati, permette l'esecuzione sia di tecniche di machine learning che di business intelligence [69].

8. La separazione tra storage e calcolo, già presente all'interno di data warehouse basato su una tecnologia cloud, permette di scalare in maniera più efficiente le risorse, poiché se vengono richieste più risorse di storage non si toccano quelle di elaborazione e viceversa [69].

9. Il data lakehouse gestisce dati in tempo reale, poiché le informazioni archiviate, non essendo sottoposte ad alcun processo ETL complesso, sono immediatamente pronte per l'analisi [69].

6.4.3 Vantaggi del data lakehouse

Una volta evidenziate le caratteristiche principali illustriamo i principali vantaggi derivanti dall'adozione di un sistema di data lakehouse, facendo un confronto con i modelli tradizionali.

Il principale vantaggio dei data lakehouse è la configurazione dei file in formato aperto, caratteristica ereditata dal data lake. Ciò consente ai dati di essere utilizzati da una moltitudine di strumenti di business intelligence, di machine learning, oltre a rimanere compatibili con nuove tecnologie che potranno emergere in futuro.

L'archiviazione di dati strutturati e non strutturati rende i data lakehouse adatti sia per supportare sistemi di business intelligence sia per applicazioni di machine learning.

Nell'architettura tradizionale i dati, una volta estratti dalle fonti, vengono caricati nel data lake in formato grezzo e poi successivamente trasformati e caricati all'interno di un data warehouse per poter fare analisi interattive. Questo approccio portava alla creazione di più copie dello stesso dato in più sistemi diversi, con conseguente aumento della complessità. Il data lakehouse, invece, è un'unica piattaforma che ospita dati immediatamente disponibili per l'interrogazione, riducendo così i tempi di risposta.

Avere meno copie sparse in sistemi diversi comporta una migliore gestione della governance e della sicurezza, infatti diventa più facile controllare i dati all'interno di un sistema solo, piuttosto che in due sistemi diversi, ed una riduzione dello spazio di archiviazione utilizzato con conseguente abbassamento dei costi.

6.5 Architettura dati nelle pipeline in Azure Data Factory

Le pipeline implementate in Azure Data Factory adottano un'architettura di tipo data lakehouse, che combina elementi tipici del data warehouse e del data lake, con l'obiettivo di coniugare performance, struttura e governance da un lato, e flessibilità e scalabilità dall'altro.

6.5.1 Componente Data Warehouse

La componente di data warehouse si concretizza nell'intero processo ELT a cui i dati vengono sottoposti nelle pipeline implementate su ADF: i file grezzi provenienti dal sistema intermedio vengono inizialmente caricati in un container di Azure Blob Storage, convertiti in formato tabellare e sottoposti a fasi di pulizia e normalizzazione (gestione di chiavi mancanti, tipologie di dato non valide, campi obbligatori assenti, ricalcolo di campi

derivati, ecc.).

L'intero flusso di data warehousing si completa con la creazione delle tabelle di auditing, che registrano le modifiche effettuate sui record, ottenute confrontando il file dell'ultimo run con quello appena ricevuto. Tipicamente includono colonne dedicate a:

- chi ha effettuato la modifica (utente, servizio applicativo);
- quando è avvenuta (data e ora di creazione, ultima modifica, eventuale cancellazione);
- cosa è cambiato (valore precedente e nuovo valore dell'attributo);
- tipo di operazione (INSERT / UPDATE / DELETE);
- note di contesto opzionali (motivo, origine della modifica, pipeline o processo).

Le tabelle di auditing costituiscono l'elemento centrale del data warehouse poiché sono il principale strumento di analisi per l'utente. In generale, le analisi possibili tramite le tabelle di auditing dipendono dal dominio applicativo, ma possono includere:

- analisi delle variazioni nel tempo (es. quanti pezzi di ricambio cambiano stato ogni mese, frequenza delle modifiche degli attributi);
- valutazioni dell'impatto delle modifiche sui modelli in uso, incrociando i dati di auditing con le relazioni tra componenti e modelli.

Queste analisi sono fondamentali per supportare le decisioni sulla gestione del ciclo di vita dei componenti.

È importante precisare che il data warehouse contiene solo i dati più recenti (ad esempio quelli dell'ultimo periodo o dell'ultimo ciclo di caricamento). Conservare dati per lunghi periodi all'interno del Data Warehouse è infatti costoso e comporta un progressivo degrado delle performance delle query. Di conseguenza, gli utenti che vi accedono possono effettuare analisi solo sui dati dell'ultimo periodo.

6.5.2 Componente Data Lake

Trascorso un determinato intervallo temporale, i dati contenuti nel data warehouse, in particolare le tabelle di auditing descritte in precedenza, vengono esportati in formato file e trasferiti nel data lake. Tale migrazione risponde all'esigenza di mantenere nel data warehouse solo i dati più recenti, ottimizzandone prestazioni e costi di gestione. Il data lake, grazie alla sua natura scalabile e alla possibilità di archiviare grandi volumi di informazioni, assume così la funzione di repository storico nel quale i dati possono essere conservati per periodi prolungati e continuare a essere sottoposti ad analisi avanzate su

intervalli temporali più estesi rispetto a quanto possibile nel data warehouse.

Attualmente, essendo il progetto ancora nella fase iniziale, la pipeline non prevede la copia automatica della tabella di auditing all'interno del data lake. Quando verrà introdotta l'attività "Copy Data", verrà copiata nel Blob Storage, che quindi fungerà inizialmente da semplice archivio centralizzato, senza trasformarsi subito in un data lake. Solo in un secondo momento sarà abilitata la funzionalità di Azure Data Lake Storage Gen2 (ADLS Gen2) tramite l'attivazione dell'Hierarchical Namespace, che renderà possibile una gestione gerarchica a cartelle e l'integrazione nativa con sistemi distribuiti (come Spark, Databricks e Synapse), per l'analisi avanzata dei dati. Senza questa opzione, lo storage può comunque fungere da archivio di dati grezzi o storici, utile per analisi di lungo periodo o per rielaborazioni, ma non è ancora assimilabile a un data lake completo in senso tecnico. I file storici risultano comunque consultabili e utilizzabili per analisi avanzate, sebbene in modo meno efficiente rispetto a un data lake nativo.

Inoltre, i file originali, ossia quelli che non hanno subito alcun tipo di modifica, vengono salvati anche in un container di backup che può essere considerato una forma di data lake. In questo modo è possibile risalire alla fonte dei dati presenti nel database, confrontare le diverse versioni ricevute e comprendere eventuali aggiornamenti, mantenendo la tracciabilità e uno storico utile per verifiche tecniche e rielaborazioni future.

6.5.3 Componente Data Lakehouse

L'integrazione tra queste due componenti definisce l'architettura di tipo lakehouse:

- per i dati recenti e più utilizzati, è disponibile il data warehouse con tabelle strutturate, query performanti e strumenti di auditing;
- per i dati storici, viene sfruttato lo storage Blob come repository a basso costo, con potenzialità di evoluzione futura verso un vero e proprio data lake (ad esempio abilitando ADLS Gen2 o introducendo layer come Delta Lake o servizi come Synapse Serverless).

In questo modo, il sistema riesce a garantire sia l'operatività quotidiana con dati aggiornati e coerenti, sia la conservazione dello storico a fini analitici, mantenendo un equilibrio tra costi, scalabilità e qualità del dato.

6.6 Architettura dati su Azure: stato attuale e sviluppi futuri

Nel nostro progetto la gestione dei dati viene affrontata con strumenti diversi, ciascuno con caratteristiche e finalità specifiche. Nei sottoparagrafi seguenti vengono presentati sia

gli utilizzi attuali sia le possibili estensioni future.

6.6.1 PostgreSQL

In questo progetto è stato adottato PostgreSQL come sistema di gestione dei dati, che permette di creare le tabelle che utilizziamo all'interno delle nostre pipeline. In questo modo i dati vengono interrogabili tramite linguaggio SQL. Oltre a questo ruolo di database relazionale, nel nostro caso PostgreSQL viene utilizzato anche come data warehouse, ossia come ambiente centralizzato in cui i dati sono raccolti e resi disponibili per attività di analisi e reporting, soprattutto quando i volumi non sono troppo grandi.

6.6.2 Azure Synapse Analytics

Quando le quantità di dati aumentano molto o quando si vogliono integrare fonti diverse (es. dati relazionali, file, log, immagini, JSON, ecc.), PostgreSQL può diventare limitato: le query possono rallentare perché lo spazio occupato cresce e in più mantenere un archivio storico nel database può diventare molto costoso. Per questo motivo, molte aziende scelgono di affiancare o sostituire PostgreSQL con soluzioni pensate apposta per la scalabilità e l'analisi di grandi volumi, ovvero Azure Synapse Analytics, che è la piattaforma cloud di Microsoft per la gestione dei dati e l'analisi. Synapse si differenzia da un database tradizionale perché è stato progettato per lavorare con grandi quantità di dati e per supportare carichi analitici più complessi.

Le caratteristiche principali di Synapse sono:

- Architettura distribuita (MPP – Massively Parallel Processing): le query vengono spezzettate e distribuite su più nodi che lavorano in parallelo, riducendo i tempi di risposta.
- Due modalità di utilizzo:
  - Dedicated SQL pool: prevede il caricamento di tutti i dati all'interno del data warehouse e ciò risulta utile quando è necessario spesso consultare ed effettuare analisi su una grande quantità di dati;
  - Serverless SQL pool: permette di interrogare i dati salvati su uno storage esterno, anche in formati CSV, Parquet o JSON, senza la necessità di caricarli nel database. In questo modo si paga solo per le query eseguite.
- Integrazione con altri servizi Azure: può collegarsi direttamente con altri strumenti già presenti in Microsoft, come ad esempio Power BI, per interrogare i dati presenti nel data warehouse senza la necessità di trasformarli, rendendo così immediata la creazione di grafici o report.

Questa flessibilità rende Azure Synapse Analytics un'ottima estensione di PostgreSQL: mentre quest'ultimo continua a gestire i dati operativi e le analisi di base, Synapse può essere utilizzato per scenari più complessi, come l'elaborazione di dati storici o l'integrazione di sorgenti eterogenee [74]. Nel nostro progetto, il data warehouse è attualmente implementato su PostgreSQL, e non è previsto, almeno nel breve termine, un piano di migrazione verso Synapse, dal momento che le stime effettuate indicano come l'architettura attuale sia sufficiente a coprire le esigenze. Tuttavia, l'adozione di Synapse può comunque essere considerata come una possibile evoluzione futura, qualora i volumi di dati o la complessità delle analisi dovessero richiederlo.

6.6.3 Azure Data Lake Storage

Il passaggio successivo riguarda l'implementazione di un'architettura basata su Data Lake, ossia Azure Data Lake Storage (ADLS), che rappresenta lo strumento di riferimento per l'archiviazione di dati eterogenei nel loro formato nativo nel lungo periodo, visti i costi contenuti. Questo approccio consente di centralizzare le informazioni e renderle disponibili per analisi successive.

Le caratteristiche principali di Azure Data Lake sono:

- Hierarchical Namespace: permette di organizzare meglio i dati, in cartelle e sottocartelle, garantendo così una gestione più semplificata dei dati anche se contenuti in archivi molto estesi, a differenza del Blob Storage, in cui tutti i file sono contenuti in un'unica grande cartella;

- Capacità di archiviazione: può memorizzare volumi molto grandi, fino a diversi petabyte, mantenendo un costo più contenuto rispetto a un database relazionale tradizionale;

- Integrazione con altri strumenti di analisi: i dati possono essere interrogati direttamente all'interno del data lake, senza la necessità di spostarli, ad esempio Synapse può eseguire query SQL sui file in ADLS. [75]

Attualmente, trovandoci ancora nella fase iniziale del progetto, i dati vengono archiviati in Azure Blob Storage, che funge da contenitore di file grezzi. Nonostante questo servizio consenta già di effettuare alcune analisi sui dati, non offre le stesse funzionalità avanzate tipiche di un data lake. Per questo motivo è previsto un passaggio verso Azure Data Lake Storage, così da disporre di uno strumento più adatto alla gestione scalabile dei dati e alla loro integrazione con servizi di analisi e machine learning.

6.6.4 Databricks

Databricks è una piattaforma che combina le funzionalità di Data Lake e Data Warehouse in un unico ambiente scalabile, noto come Data Lakehouse [73]. Questo approccio consente di gestire sia dati strutturati che non strutturati, centralizzandoli in un unico sistema che supporta analisi avanzate, machine learning e business intelligence [71].

Uno dei suoi punti di forza è l'integrazione nativa con strumenti di data engineering, data science e business analytics, che consente ai vari team di lavorare nello stesso ambiente condiviso senza dover passare da applicazioni diverse [73]. La piattaforma offre notebook interattivi per scrivere codice, visualizzare dati e costruire pipeline, facilitando così lo sviluppo di modelli di machine learning e analisi avanzate [73].

Databricks utilizza un motore di calcolo distribuito che permette di elaborare i dati in parallelo su più nodi, riducendo i tempi di esecuzione anche con dataset molto grandi [71].

Un altro elemento distintivo è Delta Lake, il livello di archiviazione open source che introduce funzionalità avanzate come transazioni ACID, gestione delle versioni dei dati (time travel) e maggiore affidabilità, migliorando la qualità delle analisi [71].

La piattaforma supporta diversi linguaggi e framework (ad esempio SQL, Python, R, Spark), rendendo possibile integrare e trasformare dati provenienti da fonti eterogenee in modo flessibile [73]. Inoltre Databricks si integra facilmente con altri servizi cloud e strumenti BI, rendendo più immediato il passaggio dai dati grezzi agli insight e ai report [73].

Conclusioni

Il lavoro presentato ha affrontato in maniera organica il tema della transizione dai sistemi mainframe alle moderne architetture cloud, delineando le motivazioni tecnologiche ed economiche che spingono le imprese a intraprendere questo percorso e mostrando, al contempo, le implicazioni pratiche che esso comporta. I mainframe, a lungo considerati l'ossatura dei sistemi informativi per la loro affidabilità, sicurezza e capacità transazionale, si trovano oggi a dover fronteggiare criticità strutturali: costi elevati di manutenzione, rigidità architetturale, difficoltà di integrazione con sistemi eterogenei e scarsità di competenze specialistiche. Parallelamente, il paradigma del cloud computing si è affermato come risposta naturale a tali limiti, introducendo un modello scalabile, flessibile e più sostenibile, sia dal punto di vista operativo che economico.

In questo contesto, Microsoft Azure si distingue come una piattaforma di riferimento, capace di offrire una vasta gamma di servizi in grado di coprire l'intero ciclo di vita dei dati: dall'integrazione con Data Factory, alla gestione tramite Data Lake Storage, fino ad arrivare alle analisi avanzate rese possibili da Synapse Analytics e Databricks. L'esperienza applicativa maturata nel progetto aziendale ha consentito di comprendere in modo concreto i vantaggi derivanti da tali strumenti. Le pipeline realizzate con Azure Data Factory hanno reso possibile un flusso dati strutturato, governato da controlli di qualità, auditing e meccanismi di recycle, permettendo di ridurre il codice custom e garantire maggiore standardizzazione. L'adozione di un'architettura dati orientata al paradigma lakehouse ha inoltre consentito di coniugare la robustezza dei data warehouse con la flessibilità dei data lake, abilitando prospettive di crescita futura verso analisi sempre più complesse e scenari di data science e intelligenza artificiale.

Dal punto di vista metodologico, la migrazione non è stata intesa come un semplice spostamento tecnico, bensì come un percorso di modernizzazione in grado di ridefinire i processi aziendali, la governance e le modalità di utilizzo delle informazioni. La scelta di un approccio "rebuild" permette di superare i vincoli ereditati dai sistemi legacy e di costruire un'infrastruttura moderna, scalabile e integrata, in grado di rispondere alle esigenze future. Il progetto ha inoltre mostrato come la riuscita di una transizione di questo tipo non dipenda solo dalla tecnologia adottata, ma anche da fattori organizzativi quali il

change management, la formazione del personale e il coinvolgimento degli utenti chiave.

Guardando agli sviluppi futuri, le prospettive di evoluzione riguardano principalmente l'architettura dati su Azure. Un primo passo sarà l'adozione di Azure Data Lake Storage Gen2, che, grazie alla struttura gerarchica e alla piena compatibilità con framework analitici come Spark, rappresenta una base solida per gestire volumi crescenti di dati. Parallelamente, il sistema attualmente basato su PostgreSQL potrà essere affiancato o sostituito da Azure Synapse Analytics, che abilita analisi su scala maggiore grazie all'elaborazione distribuita e alle capacità MPP (Massively Parallel Processing). L'introduzione di Databricks con Delta Lake rappresenterà invece un ulteriore salto evolutivo: il supporto alle transazioni ACID, al versioning dei dati e al time travel renderà possibile sviluppare modelli di machine learning, scenari di analisi predittiva e strumenti avanzati di business intelligence su basi dati storiche e complesse.

Più in generale, lo scenario futuro vede la convergenza verso architetture ibride e multicloud, in cui la possibilità di combinare servizi di diversi provider riduce i rischi di vendor lock-in e aumenta la resilienza dei sistemi. Al tempo stesso, si assisterà a un'intensificazione dei meccanismi di automazione, con pipeline sempre più intelligenti e adattive, e a una crescente integrazione con modelli di intelligenza artificiale generativa, capaci di supportare il monitoraggio, la documentazione automatica e la qualità del dato. Infine, non meno rilevante sarà l'evoluzione dei temi di governance e sicurezza: la gestione conforme al GDPR, l'adozione di modelli Zero Trust e l'impiego di strumenti di auditing avanzato saranno elementi imprescindibili per garantire continuità e affidabilità in un contesto sempre più complesso e distribuito.

In conclusione, la transizione dal mainframe al cloud non rappresenta un punto di arrivo, bensì un processo in continua evoluzione. Il progetto qui discusso costituisce un tassello di un percorso più ampio, in cui le imprese sono chiamate a trasformare non soltanto le proprie infrastrutture tecnologiche, ma anche la cultura organizzativa e le modalità di gestione delle informazioni. Il cloud, e in particolare l'ecosistema Azure, si configura come piattaforma abilitante di questa trasformazione, in grado di accompagnare le aziende verso un modello di innovazione continua, sostenibilità e crescita.

Ringraziamenti

Un pensiero speciale va a tutte le persone che hanno reso possibile questo lavoro.
Ringrazio il mio relatore Prof. Apiletti per la disponibilità e per l'opportunità concessami nello sviluppo del presente elaborato.
Un grazie sincero va ai miei tutor aziendali, Simone, Matteo e Laura, per la loro guida e per aver condiviso con me competenze ed esperienza durante questo percorso.
Un ringraziamento profondo e infinito ai miei genitori, che con amore, dedizione e pazienza mi hanno sempre sostenuta, senza mai esercitare pressioni, ma offrendo la loro presenza costante. Sono stati il mio punto fermo, la mia forza e il pilastro che ha reso possibile arrivare fin qui.
Un pensiero colmo di gratitudine va infine a mia sorella, che mi ha accompagnata lungo questo cammino, standomi vicino nei momenti più difficili e condividendo con me quelli più felici, sempre con pazienza e premura.

Bibliografia

[1] IBM, Mainframe: cos'è e come funziona, IBM, 2024. [Online]. Available: https://www.ibm.com/it-it/topics/mainframe

[2] Fastweb, Cosa è un mainframe, Fastweb Digital Magazine, 2023. [Online]. Available: https://www.fastweb.it/fastweb-plus/digital-magazine/cosa-e-un-mainframe/

[3] Precisely, Mainframe History: From the Beginning to Today, Precisely Blog, 2023. [Online]. Available: https://www.precisely.com/blog/mainframe/mainframe-history/

[4] Kyndryl, The modern role of mainframes in hybrid cloud, Kyndryl, 2023. [Online]. Available: https://www.kyndryl.com/us/en/services/core-enterprise-and-zcloud

[5] IBM Newsroom, IBM z16 with Telum Processor, IBM, Apr. 5, 2022. [Online]. Available: https://newsroom.ibm.com/2022-04-05-IBM-Announces-z16

[6] Uitgelegd.net, Perché i sistemi mainframe sono limitati, 2023. [Online]. Available: https://uitgelegd.net/it/perche-i-sistemi-mainframe-sono-limitati

[7] Straits Research, Mainframe Market Report 2024–2032, Straits Research, 2024.

[8] Mordor Intelligence, Mainframe Market Size & Trends 2023–2033, Mordor Intelligence, 2023.

[9] MarketsandMarkets, Mainframe Modernization Market – Forecast 2025–2030, MarketsandMarkets, 2024.

[10] TechRadar, IBM z17, il mainframe per l'era dell'AI e del cloud ibrido, TechRadar, 2025.

[11] TechRadar, Generative AI: a game changer for mainframe modernization, TechRadar, 2024.

[12] Penn Engineering, ENIAC – History and Heritage. University of Pennsylvania. [Online]. Available: https://www.seas.upenn.edu/about/history-heritage/eniac/

[13] History.com Editors, UNIVAC, the first commercially produced digital computer, is dedicated, History.com. [Online]. Available: https://www.history.com/this-day-in-history/june-14/univac-computer-dedicated

[14] Unsplash, Unsplash. [Online]. Available: https://unsplash.com

[15] IBM Developer, A tour inside the IBM z16. IBM, 2022. [Online]. Available: https://developer.ibm.com/blogs/a-tour-inside-the-ibm-z16

[16] TechRadar, IBM z17, il mainframe per l'era dell'AI e del cloud ibrido. TechRadar, 2025. [Online]. Available: https://www.techradar.com

[16] TechRadar, IBM z17, il mainframe per l'era dell'AI e del cloud ibrido. TechRadar, 2025. [Online]. Available: https://www.techradar.com

[17] M. N. O. Sadiku, S. M. Musa, and O. D. Momoh, "Cloud computing: Opportunities and challenges," IEEE Potentials, vol. 33, no. 1, pp. 34–36, Jan.–Feb. 2014, doi: 10.1109/MPOT.2013.2279684.

[18] F. Magoulès, J. Pan, and F. Teng, Cloud Computing: Data-Intensive Computing and Scheduling. Boca Raton, FL, USA: CRC Press, Taylor & Francis Group, 2013, ISBN 978-1-4665-0783-8.

[19] L. Qian, Z. Luo, Y. Du, and L. Guo, "Cloud computing: An overview," in Proc. 1st Int. Conf. Cloud Computing (CloudCom 2009), Lecture Notes in Computer Science, vol. 5931, M. G. Jaatun, G. Zhao, and C. Rong, Eds. Berlin, Heidelberg: Springer, 2009, pp. 626–631.

[20] S. Heng and S. Neitzel, Cloud Computing: Clear Skies Ahead. Frankfurt am Main, Germany: Deutsche Bank Research, Mar. 2012. [Online]. Available: https://www.researchgate.net/publication/249796971

[21] A. Rajaraman, "Cloud computing," Resonance: Journal of Science Education, vol. 19, no. 3, pp. 242–258, Mar. 2014.

[22] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, "The characteristics of cloud computing," in Proc. 2010 39th Int. Conf. Parallel Processing Workshops (ICPPW), San Diego, CA, USA, 2010, pp. 275–279, doi: 10.1109/ICPPW.2010.45.

[23] IBM, Data Centers. IBM Think, 2024. [Online]. Available: https://www.ibm.com/it-it/think/topics/data-centers

[24] IBM, Virtualization. IBM, 2024. [Online]. Available: https://www.ibm.com/it-it/topics/virtualization

[25] GeeksforGeeks, Difference between Cloud Computing and Grid Computing. GeeksforGeeks, 2023. [Online]. Available: https://www.geeksforgeeks.org/difference-between-cloud-computing-and-grid-computing/

[26] GDPR.eu, What is GDPR, the EU's new data protection law?. GDPR.eu, 2024. [Online]. Available: https://gdpr.eu/what-is-gdpr/

[27] A. Rashid and A. Chaturvedi, "Sufficient comparison among cloud computing services: IaaS, PaaS, and SaaS: A review," International Journal of Computer Science and Information Technologies (IJCSIT), vol. 5, no. 6, pp. 5863–5866, 2014.

[28] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generation Computer Systems, vol. 25, no. 6, pp. 599–616, Jun. 2009.

[29] Cloud Security Alliance, Zero Trust Is Not Enough: Evolving Cloud Security in 2025. CSA Blog, Apr. 17, 2025. [Online]. Available: https://cloudsecurityalliance.org/blog/2025/04/17/zero-trust-is-not-enough-evolving-cloud-security-in-2025

[30] J. Kindervag et al., Zero Trust Architecture (SP 800-207). NIST, 2020. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-207/final

[31] ENISA, Artificial Intelligence Cybersecurity Challenges. European Union Agency for Cybersecurity, 2021. [Online]. Available: https://www.enisa.europa.eu/publications/artificial-intelligence-cybersecurity-challenges

[32] NEXTDC, The Rise of Hybrid and Multi-Cloud Computing Architecture. NEXTDC Blog, 2025. [Online]. Available: https://www.nextdc.com/blog/the-rise-of-hybrid-and-multi-cloud-computing-architecture

[33] S. Khan, R. Kumar, and A. Gani, "Cyber resilience and cyber security issues of intelligent cloud computing systems," Future Generation Computer Systems, vol. 101, pp. 747–763, Dec. 2019.

[34] Open Networking Foundation, Open Networking Foundation Formed to Speed Network Innovation. Press Release, Mar. 21, 2011. [Online]. Available: https://opennetworking.org

[35] Cloud Native Computing Foundation, Cloud Native Technology Landscape & History, 2015–2024. [Online]. Available: https://www.cncf.io

[36] CogentInfo, The Evolution of Cloud Computing: Trends and Emerging Technologies Shaping 2025. Feb. 2025. [Online]. Available: https://cogentinfo.com

[37] J. Alonso et al., "Understanding the challenges and novel architectural models for multi-cloud application development," Journal of Cloud Computing, vol. 12, no. 1, 2023.

[38] Oracle, "What Is Multicloud? Uses and Benefits", Oracle Official Website, 2025.

[39] Dataversity, "A Brief History of Cloud Computing", Dataversity, 2021. [Online]. Available: https://www.dataversity.net/brief-history-cloud-computing/

[40] GPU Server Case, Edge Compute vs. Cloud Computing: Key Differences Explained. GPU Server Case Blog, 2024. [Online]. Available: https://gpuservercase.com/it/blog/edge-compute-vs-cloud-computing-key-differences-explained

[41] Microsoft, Why Azure? — Benefits of Microsoft Azure, 2025. [Online]. Available: https://azure.microsoft.com

[42] Multishoring, Top Benefits of Microsoft Azure for Businesses, 2024. [Online]. Available: https://multishoring.com

[43] Microsoft, Azure Hybrid Solutions – Architecture Center. [Online]. Available: https://learn.microsoft.com/en-us/azure/architecture/hybrid/

[44] Microsoft Azure, Global Infrastructure, 2025. [Online]. Available: https://azure.microsoft.com/en-us/explore/global-infrastructure

[45] Svitla Systems, Cloud Platforms Comparison: AWS vs Azure vs Google Cloud. Svitla Blog, Aug. 11, 2025. [Online]. Available: https://svitla.com/blog/cloud-platforms-comparison/

[46] BMC Software, AWS vs Azure vs Google Cloud: Key Differences. BMC Blogs, Jul. 18, 2024. [Online]. Available: https://www.bmc.com/blogs/aws-vs-azure-vs-google-cloud-platforms/

[47] Google Cloud, Global Locations, 2025. [Online]. Available: https://cloud.google.com/about/locations

[48] Amazon Web Services, Global Infrastructure, 2025. [Online]. Available: https://aws.amazon.com/about-aws/global-infrastructure

[49] Dev4Side, Microsoft Azure. Dev4Side Blog, 2025. [Online]. Available: https://www.dev4side.com/blog/microsoft-azure

[50] Amazon Web Services, AWS Prescriptive Guidance – Large Migration Guide (Assess, Mobilize, Migrate & Modernize, Operate). AWS, 2025.

[51] Microsoft, Cloud Adoption Framework: Migration Planning and Execution. Microsoft Learn, 2025. [Online]. Available: https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/migrate

[52] Swimm, Mainframe to Cloud Migration: 5 Approaches & 5 Critical Best Practices. Swimm.io, 2025. [Online]. Available: https://swimm.io/learn/mainframe-modernization/mainframe-to-cloud-migration-5-approaches-5-critical-best-practices

[53] A. M. A. et al., Solving the Mainframe Application Conundrum. White Paper, 2024.

[54] Overcast, Mainframe to Cloud: A Full Guide. Overcast Blog, 2025. [Online]. Available: https://overcast.blog/mainframe-to-cloud-a-full-guide-fd147c6491fd

[55] A. Al-Ali et al., "Challenges in migrating legacy software systems to the cloud—An empirical study," Journal of Systems and Software, vol. 174, p. 110890, Apr. 2021.

[56] P. Jamshidi, A. Ahmad, and C. Pahl, "Effort estimation of large-scale enterprise application mainframe migration projects: A case study," Journal of Cloud Computing, vol. 6, no. 1, pp. 1–16, 2017.

[57] Ocient, Dun & Bradstreet Case Study. Ocient, 2025. [Online]. Available: https://ocient.com/case-studies/dun-bradstreet-case-study

[58] NTT Data, Consumer Goods Company Modernizes with AWS Mainframe Migration. NTT Data, 2025. [Online]. Available: https://us.nttdata.com/en/case-studies/consumer-goods-company-modernizes-with-aws-mainframe-migration

[59] Ensono, SSAB Modernizes from Mainframe to Azure Cloud with Ensono. Ensono, 2025. [Online]. Available: https://www.ensono.com/results/client-stories/ssab-modernizes-from-mainframe-to-azure-cloud-with-ensono

[60] Microsoft Learn, Integration Runtime in Azure Data Factory. [Online]. Available: https://learn.microsoft.com/en-us/azure/data-factory/concepts-integration-runtime

[61] M. Mukesh, Understanding Integration Runtime in Azure Data Factory: A Quick Guide. Medium, 2024. [Online]. Available: https://medium.com/@mukesh_44692/understanding-integration-runtime-in-azure-data-factory-a-quick-guide-0f40a60e7fbd

[62] Microsoft Learn, Pipelines and Activities in Azure Data Factory. [Online]. Available: https://learn.microsoft.com/en-us/azure/data-factory/concepts-pipelines-activities

[63] Microsoft Learn, Pipeline Execution and Triggers in Azure Data Factory. [Online]. Available: https://learn.microsoft.com/en-us/azure/data-factory/concepts-pipeline-execution-triggers

[64] M. Mukesh, Understanding Linked Services and Datasets in Azure Data Factory. Medium, 2024. [Online]. Available: https://medium.com/@mukesh_44692/understanding-linked-services-and-datasets-in-azure-data-factory-6f1f2c

[65] D. Mazumdar, J. Hughes, and J. Onofré, The Data Lakehouse: Data Warehousing and More. arXiv preprint arXiv:2310.08697, 2023. [Online]. Available: https://arxiv.org/abs/2310.08697

[66] M. Cherradi and A. El Haddadi, "Data Lakehouse: Next Generation Information System," Seminars in Medical Writing and Education, vol. 3, p. 67, 2024, doi: 10.56294/mw202467.

[67] A. Nambiar and D. Mundra, "An overview of data warehouse and data lake in modern enterprise data management," Big Data and Cognitive Computing, vol. 6, no. 4, p. 132, 2022, doi: 10.3390/bdcc6040132.

[68] IBM, Data Warehouses vs. Data Lakes vs. Data Lakehouses. IBM Think, 2024. [Online]. Available: https://www.ibm.com/think/topics/data-warehouse-vs-data-lake-vs-data-lakehouse

[69] B. J. Mary, Unified Data Architecture for Machine Learning: A Comparative Review of Data Lakehouse, Data Lakes, and Data Warehouses, Apr. 2025. [Online]. Available: https://www.researchgate.net/publication/390533115

[70] L. MacDonald, 5 Layers of Data Lakehouse Architecture Explained. Monte Carlo Data Blog, Jan. 05, 2024. [Online]. Available: https://www.montecarlodata.com/blog-data-lakehouse-architecture-5-layers/

[71] M. Armbrust, A. Ghodsi, R. Xin, and M. Zaharia, "Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics," in Proc. 11th Conf. Innovative Data Systems Research (CIDR'21), Online, Jan. 11–15, 2021. [Online]. Available: https://www.cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf

[72] IBM, Che cos'è il Regolamento generale sulla protezione dei dati (GDPR). IBM Cloud, 2024. [Online]. Available: https://www.ibm.com/it-it/cloud/compliance/gdpr-eu

[73] Databricks, What is a Data Lakehouse?. Databricks Glossary, 2025. [Online]. Available: https://www.databricks.com/glossary/data-lakehouse

[74] Microsoft Learn, Copy and Transform Data in Azure Database for PostgreSQL Using Azure Data Factory or Synapse Analytics. [Online]. Available: https://learn.microsoft.com/en-us/azure/data-factory/connector-azure-database-for-postgresql

[75] V. Loghin, Azure Synapse Analytics Integrate Data Lake. Medium, 2021. [Online]. Available: https://medium.com/@valentin.loghin/azure-synapse-analytics-integrate-data-lake-2608a5c5005f

[76] Microsoft Learn, Microsoft Global Network – Azure, Microsoft Learn, 2025. [Online]. Available: https://learn.microsoft.com/en-us/azure/networking/microsoft-global-network

[77] AnsiByteCode, "Azure vs AWS vs Google Cloud: A Comprehensive Comparison," AnsiByteCode Blog, 2024. [Online]. Available: https://ansibytecode.com/azure-vs-aws-vs-google-cloud-a-comprehensive-comparison/