Technological Infrastructures

Abstract

L'obiettivo del corso è forninre conoscenza solida delle piattaforme tecnologiche e computing platforms. Differenza delle responsabilità di Data Scientist e Data Engineer. Il corso è diviso in 2 parti. L'esame consiste in 15 domande sia aperte che chiuse per parte e si può arrivare a 30 con lo scritto, si può fare un progetto non obbligatorio che vale al più 3 punti.

1 Componenti di NIST

NIST sviluppa standard di riferimento per il pubblico. Software Architecture è un organizzazione globale di sistemi software, quindi consiste in:

- divisione della componenti software in sottosistemi;
- Definisce le politiche con cui questi sistemi interagiscono;
- definisce le interfacce tra le varie componenti.

Un'architettura di riferimento è essenzialmente un template (scatola vuota con elementi prefissati), fornisce solo il vocabolario usato comunemente per discutere le implementazioni di un dato software.
Un'architettura di riferimento per il software non è altro che architettura software dove le strutture e i vari elementi e relazioni sono forniti dai template.

NIST fornisce l'architettura di riferimento per i Big Data che:

- Fornisce un linguaggio comune per i stakeholders;
- Incoraggia aderenza ai standard comuni;
- Permette di implementare le architetture con una certa consistenza;
- Illustra e migliora la comprensione delle componenti, processi e sistemi di Big Data;

1.1 I 5 ruoli principali per Big Data

L'architettura concettuale dei Big Data è un architettura a croce con due assi: Information value (IV) e Information Technology (IT) (Fig 1).
I 5 ruoli principali sui 2 assi dei Big Data abbiamo:

1.1.1 System Orchestrator

Il system orchestrator coinvolge spesso anche Information Value chain, poiché si preoccupa di implementare e monitorare i processi business a livelli enterprise e le varie politiche sui dati: Redere i dati accessibili per un tempo limitato oppure fornire i dati a velocità diversa(passando il dato in memoria al disco). Può assegnare/fornire componenti framework fisici o virtuale al sistema, questa assegnazione può essere molto spesso elastica ed indipendente. Può fornire supporto GUI e collegare i varie applicazioni a un livello alto, e attraverso il manage-

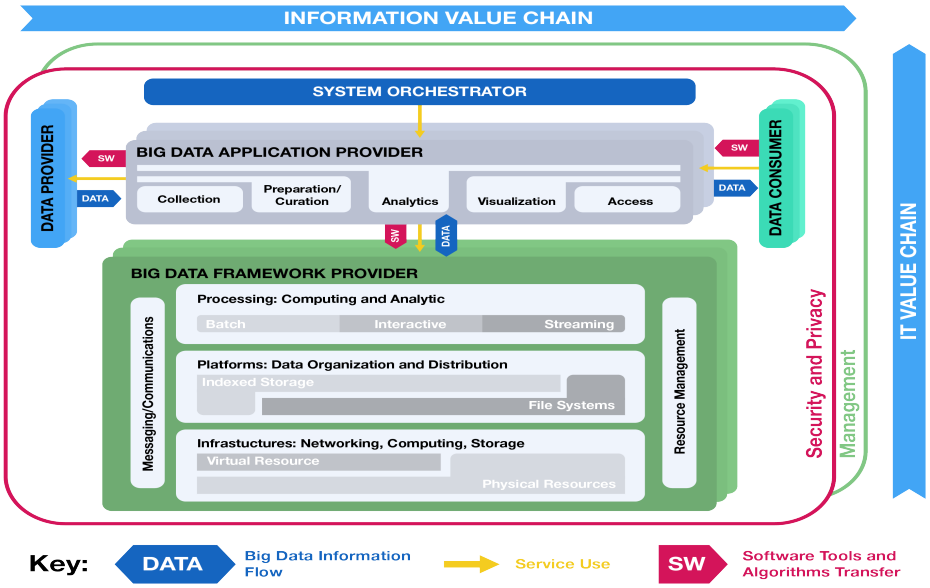

Figure 1: NBDRA Conceptual Model

ment fabbric monitorare i carichi e il sistema per garantire/specificare la qualità del servizio necessaria per i vari carichi. E' molto spesso centralizzato. Es. Ambari/Cloudera

1.1.2 Data Provider

Può essere sia un software (ad es. in una pipeline più grande) che una persona. Se è una persona metterà i suoi dati e userà gli strumenti di collection e curation/preparation per caricare i dati sul sistemi e migliorarne la qualità, Se un software metterà is suoi dati a disposizione attraverso delle interfacce come Apache Scoop. Il data provider può essere interno o esterno alla piattaforma, deve fornisce i diritti di accesso ai dati, ed è obbligato a seguire le policy di privacy e security fabbric. I dati possono essere inseriti in pull o push. es. Flume per caricare i dati da MySQL a HDFS.

1.1.3 Data Consumer

Riceve i output dei sistemi BigData, può anche lui fare pull e push dei dati, può usarli le informazioni per data reporting, retrival/search e visualization. Ci deve essere l'autenticazione ed autorizzazione daparte della privacy and security fabbric per la comunicazione tra l'architettura e il Data Consumer.

1.1.4 Big Data Application Provider

Corrisponde alle attività tipiche di un Data Scientist, che esistono anche nei sistema tradizionali ma hanno delle trasformazioni nella implementazioni con i Big Data. Queste attività sono:

- Collection: Si occupa di gestire l'interfaccia fornita dal Data Provider, salva/gestisce questi dati in una certa zona affinché questi non vengano persititi, inoltre implementa

funzionalità di estrazione dati dal data provider;
- Preparation: Effettua Data Validation, rimozione outlier, standardizzazione, formattazione e arricchimento. Cerca di promuovere dati di alta qualità;
- Analytics: Estrazione conoscenza dai dati, sfruttando il software sottostante del Big Data Framework Provider;
- Visualization: Presentazione dati in maniera visuale;
- Access: E' l'opposto della collection, si preoccupa di esporre i dati verso l'esterno.

1.1.5 Big Data Framework Provider

Fornisce le infrastrutture per supportare i Big Data Application Provider. Si occupa in particolare di:

- Processing dei dati - ha una dualità: da una parte è un framework che mette a disposizione delle interfacce per la programmazione per fare certe cose (es. MapReduce) e dall'altra parte definisce come l'implementazione viene effettuata. I framework possono variare tra un proccessamento batch e in streaming.
- Platforms per l'organizzazione e immagazzinamento dei dati - può contenere i meta-dati insieme alle descrizione semantiche dei dati. Può essere relazionale distribuito o non relazionale.
- Infrastructures per l'esecuzione fisica del nostro software, è l'insieme delle risorse computazionali fisiche o virtuali sulle quali il nostro sistema Big Data gira, può essere costituito da server di grandi o piccole dimensione. Queste componenti forniscono:
  - Networking - Possono essere definiti attraverso software e possono essere reti fisiche che può essere a sua volta partizionato in reti virtuali. Possono essere reti puramente virtualizzate cioè tutto quanto (firewall, router, load balancing) sono realizzate in maniera virtuale (es. VM dentro la nostra macchine);
  - Computing - Hardware, Software, OS, memoria per il computing;
  - Storage - dischi per storare (in locale), RAID, in rete ecc.
  - e altri servizi come il Raffrendamento, l'apparto elettrico e la sicurezza

  Può essere deployato su ambienti fisici che virtualizzati (nativi, hostati o contenerizzati).

Inoltre ha 2 ruoli diffusi nelle 3 componenti sopraindicate:

- Comunicazione e messaggistica tra le componenti;
- Gestione delle risorse per l'integrazione delle componenti.

1.2 I 2 ruoli diffusi per Big Data

questi 2 ruoli prendono il nome di Fabric, il termine Fabric(tessuto) viene usato perché queste 2 ruoli cross-cutting, cioè sono presenti un pò ovunque nell'architettura.

1.2.1 Management fabbric

Le due attività principali associate:

- Gestione del sistema per provvedere le risorse, gestione dei software e pacchetti e infine la gestione delle configurazioni e performance delle varie pipeline;
- Ciclo di vita dei Big Data, BDLM (Big Data Life Cycle Management), contiene l'enforcing delle Policy (es. encod-

ing/decoding), gestione dei meta data (data governance), accessibilità dei dati, data recovery e la loro preservazione.

1.2.2 Security and Privacy Fabric

Si occupa delle tre caratteristiche tipiche delle security:

- Autenticazione: Indica tutte le attività che validano l'utente;
- Autorizzazione: Una volta autenticato l'utente verifica i suoi permessi, es. alcuni dati potrebbero non essere accessibili a certi utenti;
- Auditing: riguarda la registrazione degli eventi che accadono nel sistema, può far partire un allarme in caso di evento anomalo o a posteriori analizzare la sequenza di eventi (con i file log).

2 Virtualizzazione

Per i computer sono stati definiti con le 5 componenti classiche:

1. Input Devices: Tastiera ecc.
2. Output Devices: Display ecc.
3. Storage Devices: Volatile(RAM), Permanente(HD, SSD)
4. Processore:
    - Datapath
    - Control
5. Network

La virtualizzazione permette l'esecuzione di più sistemi operativi simultaneamente sulla macchina in maniera totalmente isolata. Può essere visto come una emulazione di un software o hardware su cui altri software possono eseguirsi, questo ambiente emulato è detto virtual machine. Il concetto di VM è stato sviluppato negli anni 60 da IBM sui mainframes. Viene abbandonato con la nascita di PC moderni e ripreso con la crescita recente di cloud. La virtual machine viene ottenuto attraverso un Virtual Machine Monitor detto anche ipervisore.
L'**Hypervisor**(Ipervisore) è un software che giace sotto gli OS virtualizzati per offrire le funzionalità di condivisione delle risorse disponibili in modo tale che il programma o OS in esecuzione veda queste risorse come se fosse a lui dedicate. Le risorse sono CPU, memoria, storage e la rete.
Vi sono diversi benefici della virtualizzazione:

- Visione unificata delle risorse es. vedo tanti dischi come un unico disco;
- Consolidazione delle risorse virtualizzate, in modo da avere l'ottimo utilizzo delle risorse;
- Facilità di implementare la ridondanza per copiare gli ambienti virtualizzati;
- Facilità la migrazioni di sistema su un altro, inoltre se non cambio ipervisore la macchina(virtuale) funzionerà identicamente a prima;
- Gestione centralizzata del hardware e software.

Altri benefici/proprietà della virtualizzazione sono:
Workload Isolation: attraverso la virtualizzazione è possibile isolare completamente i programmi, che ha miglioramenti anche nella sicurezza, inoltre aumenta affidabilità poiché il fallimento di un programma non comporta fallimento dei programmi poiché sono isolati, inoltre si risolvono anche i problemi riguardanti i conflitti di librerie in questo modo. Infine si ottiene un controllo sulle performance poiché l'esecuzione di una VM non affligge il performance dell'altra;
Workload Migration: ciò aiuta in:

- Mantenimento di Hardware;

- Load Balancing;
- Fault Tolerance;
- Disaster Recovery.

Poiché possiamo spostare tutto l'ambiente virtualizzato su una nuova macchina in maniera abbastanza trasparente, per fare ciò la macchina dovrebbe essere sospesa, totalmente serializzata per essere inviata nella rete, migrata su una nuova macchina e fatta ripartire immediatamente o solo salvata senza esecuzione.

Consolidazione: Sfruttando il Workload Migration è possibile consolidare macchine separate su un unica piattaforma riducendo i costi (usata molto spesso nei Datacenter in orari non di picchi).

I diversi tipi di HyperVisor sono (Fig 2):

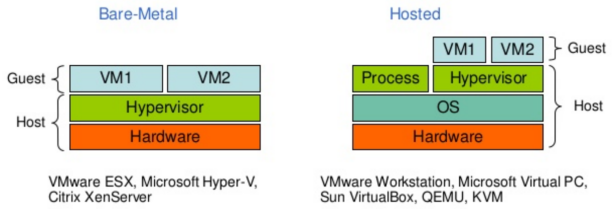

Figure 2: I tipi di Hypervisor

- **Hosted**: in questo caso ipervisore è un processo che gira al di sopra del sistema operativo e permette l'esecuzione di più macchine Guest. In questo caso quindi bisogna installa prima un OS su cui verrà installa VMM (ipervisore) e adesso l'host potrà eseguire le applicazioni all'interno della sua finestra. Il vantaggio qua è la facilità di installazione e configurazione, inoltre la HostOS e GuestOS sono non modificati e non dipendono dal particolare hardware, ma gli svantaggi sono la degradazione delle performance e la mancanza di supporto real time OS poiché vi sono varie entità/software in mezzo;
- **Bare-Metal**: in questo caso ipervisore funziona direttamente al di sopra del Hardware. In questo caso l'ipervisore è un OS molto leggero e comunica direttamente con hardware al posto di dipendere su un altro OS. I vantaggi sono miglioramento in I/O e supporto real time, mentre gli svantaggi sono la difficoltà di istallazione e configurazione e la dipendenza dal tipo di hardware specifico.

Ci sono principalmente 2 tecniche di virtualizzazione: **Software Virtualization** (di cui abbiamo parlato fino ad adesso) e **Hardware Assisted Virtualization**.

Nella **Virtualizzazione Totale** VMM si preoccupa di emulare in maniera completa tutto l'hardware, quindi avremo un processore, memoria, disco e network virtuale. in questo caso OS ospite non è consapevole dell'esistenza dell'ambiente virtuale e ogni macchina è del tutto indipendente. A livello CPU, avviene la traduzione binaria: Questo avviene in diversi anelli di sicurezza, in particolare sono 4, dove l'anello 0 è quello più privilegiato e permette l'esecuzione del codice direttamente sul hardware e qua dove viene eseguito il kernel dell'OS. Le applicazione dell'utente vengono eseguite sull'ultimo anello (anello 3).

L'ipervisore gira sull'anello 0 mentre le GuestOS gira sull'anello 1, quindi hanno più permessi delle normali applicazioni. VMM ha accesso sul anello 0 per avere accesso diretto sulla CPU piuttosto che virtualizzarla.

La **Para-Virtualizzazione** ha un approccio diverso rispetto alla virtualizzazione totale (è una via di mezzo tra total virtualization e bare-metal), in questo caso il Guest sono consapevoli della VMM e usa chiamate speciali in alcuni casi per essere eseguite direttamente sul hardware e ciò comporta un miglioramento nella perfor-

mance ma ciò lo rende meno flessibile.
La **OS-level Virtualization (Containerization)** non usa la VMM, la virtualizzazione è fornita direttamente dal HostOS che esegue tutte le funzioni di un ipervisore totalmente virtualizzato, quindi ha una partizione puramente virtuale delle risorse, e ciò comporta un'assegnazione flessibile delle risorse alle varie applicazioni. Es. Docker.
Ci sono 3 modelli di servizio:

1. **Server Virtualization**: supponiamo di avere diversi server su macchine diverse, in caso di un problema (crash di un nodo) o vi è un bisogno di upgrade, in caso di macchine fisiche dovrò prendere lo stesso modello o lo stesso vendore, la soluzione è quella di sfruttare una medesima macchina con un virtualizzatore e mettere i diversi server insieme. In questo modo ho una consolidazione, risorse condivise, una gestione centralizzata, facilità di migrazione, maggiore ROI e meno spazio occupato. La Disaster Ricovery e scalabilità viene facilitata, diversi modelli a scelta (basta che sia uguale l'ipervisore), ho maggiore disponibilità.
2. **Desktop Virtualization**: è la tecnologia che separa l'ambiente desktop e le applicazione software dal cliente fisico che lo usa. Il nostro desktop diventa un thin client che usa PC privo della memoria RAM e potenza calcolo necessaria per far funzionare una macchina reale, la macchina gira su un pool di VM su un server su un data center.
   I benefici sono molteplici: Upgrade di software e OS facilitato, alta disponibilità, fault tolerance, accessibile da LAN, WAN, Internet, inoltre vi è la possibilità di far eseguire una piccola parte della computazione dal dispositivo locale.
3. **Application Virtualization**: Molto simile alla Desktop virtualizzazione, solo che al posto di tutto il desktop, sono le applicazione ad essere in virtualizzate, quindi un applicazione che gira sul desktop, fa la computazione sul cloud, una piccola parte della computazione può essere effettuata anche in locale. I vantaggi sono molto simili a quelli del Desktop Virtualization, un vantaggio aggiuntivo può essere che pago solo quello che uso, riducendo i costi delle licenze.

3 Cloud

Definizione "vera" del cloud da ricordare: è un tipo di servizio distribuito di sistemi interconnessi e computer virtualizzati dinamicamente provvisionati e presentati come un unica risorsa computazionale basato su service-level agreement.
Sostanzialmente il cloud è la possibilità di avere accesso alle risorse computazionale attraverso la rete on-demand ed è composto da 5 punti chiave, 3 modelli di servizio e 4 modelli di deployement.

3.1 The 5 Properties of cloud

Le 5 proprietà aggirano attorno ad una idea centrale del cloud: Utility Computing, SOA (Service Oriented Architecture) + SLA(Service Level Agrement).
Utility Computing nel senso che il service provider fornisce le risorse computazionali e le infrastrutture che servono al cliente e li fa pagare in base all'utilizzo piuttosto che con un costo fisso, come i servizi on-demand per massimizzare l'uso efficiente, minimizzando i costi nella maniera meno trasparente possibile(Senza mostrare al cliente tutto ciò che si fa).
SOA (Service Oriented Architecture): è un

insieme di servizio che comunicano tra di loro poiché non basta di solito un solo servizio per implementare un servizio web completo per utente.
SLA (Service Level Agreement): è un contratto tra il service provider e il cliente che specifica il livello di servizio che il service provider deve fornire in termini di QoS. Il **QoS**(Quality of Service) è un set di tecnologie per gestire il traffico della rete in modo da migliorare la user experience, ormai è associato a un significato più generico riguardante le valutazioni di tecnologie non funzionali, cioè non mi interessa solo la funzionalità (il risultato della mia richiesta) ma anche la sua efficienza.
Le metriche più comuni delle SLA sono up-time e down-time, Response time. Se le metriche garantite non vengono rispettato vi sono delle penalità sui service provider.
La tecnologia grazie al quale cloud funziona sono:

- Hardware Virtualization;
- Computing distribuito e parallelo;
- Service-oriented Computing;
- Autonomic Computing (reazione a cambiamenti del sistema).

Le 5 **proprietà/caratteristiche** che identificano il cloud sono:

1. **Scalabilità, Elasticità**:
   - Scalabilità è una proprietà del sistema di crescere in maniera graduale col crescere di richieste, può essere orizzontale (Scale In & Scale Out) o verticale (Scale Up & Scale Down).
   - Elasticità è l'abilità di adattarsi automaticamente per inizializzare la scalabilità.

   Ciò si può ottenere attraverso provisioning dinamico, che fa riferimento è un ambiente complesso che permette la allocazione e deallocazione on-demand delle istanze/risorse attraverso applicazione o un console amministrativo. Di solito vengono messe delle regole per automatizzare il provisioning, ottenendo così un abbassamento dei costi e performance migliorata.
2. **Disponibilità, Affidabilità**:
   - Availibility è il rapporto tra il up-time e tempo totale di esecuzione;
   - Affidabilità è la capacità di funzionare in situazioni particolari (rottura di un nodo, attacco di hacker) per un certo tempo.

   Questi punti possono essere ottenuti attraverso sistemi di:
   - Fault Tolerance è la capacità del sistema di continuare ad funzionare anche dopo un fallimento di uno dei suoi componenti, spesso il servizio viene degradato proporzionalmente alla gravità del fallimento ma il servizio continua ad funzionare.
     Le caratteristiche principali sono che non ha un singolo punto di fallimento, la componente fallita viene individuata ed isolata per prevenire la propagazione del fallimento;
   - Resilience è l'abilità di offrire e mantenere un livello di servizio accettabile anche dopo un fallimento, essenzialmente ritornare allo stato di funzionamento dopo un caso di fallimento del sistema (ad es. viene a mancare l'elettricità). Quindi i sistemi devono avere le policy e procedure per il recupero, ad es. Backup (dei dati off-site oppure di tutto il sistema) oppure preparazione contro fault come un una corrente non-interrompibile (UPS) oppure sbalzi di corrente/tensione.
   - Sicurezza La sicurezza riguarda l'impiego di politiche e tecnologie e sistemi di controllo per protegge applicazioni e infrastrutture da accessi malevoli, quindi abbiamo il suo impiego in:
     – la protezione i dati, mantenendo

l'accesso ai dati riservato solo in base ai privilegi;
   - Gestione dell'identità per garantire l'accesso alle risorse in base ai loro privilegi (protezione dei dati);
   - Sicurezza dell'applicazione riguarda la possibilità di blindare le nostre applicazioni per accessi malintenzionati;
   - Privacy per oscurare i dati in base alle leggi privacy e gestiti solo da utenti competenti.
3. **Gestibilità, Interoperabilità**: La Gestbilità riguarda la gestione di questi sistemi attraverso un singolo punto di accesso mentre la interoperabilità è una proprietà di un prodotto o sistema per lavorare con altri prodotti/sistemi. Lo si ottiene attraverso **System control automation** e **System State Monitoring** che è un processo che monitora lo stati dei hardware, metriche dei performance, i log dei sistemi, il pattern dei accessi alla rete ecc. E' anche collegato al sistema di Billing, poiché gli utenti pagano ciò che usano e il cloud provider deve registrare le risorse/servizio usate da ciascun utente.

4. **Accessibilità, Portabilità**
5. **Ottimizzazione, Performance** Il **Load Balancing** è una tecnica per la distribuzione del workload su un sistema di più computer, CPU, HD ecc. per ottenere l'utilizzo ottimale, migliorando così: l'utilizzo del sistema, la performance e l'efficienza energetica riducendo overload.
   Il **Job Scheduler** è un applicazione che ha il compito di eseguire i processi in background (chiamati anche processi batch). Nel cloud i task intensivi computazionalmente o task che crescono dinamicamente vanno pianificato con Job Scheduler.

Da un punto di vista dell'utente, lui non vuole sapere come e cosa viene fatto né chi gestisce il servizio, ma vuole solo un servizio funzionale.

3.2 The 3 Service Model of cloud

Iniziamo con i tre modelli del servizio del cloud sono:

1. **IaaS (Infrastructure as a Service)**: es. le macchine virtuali, in questo caso il provider/vendor gestisce la virtualizzazione, il cliente ottiene le risorse virtuali, di solito da un catalogo che contiene le specifiche hardware (virtualizzato) pre-selezionate dal provider. Nella maggior parte dei casi gli OS sono anche prefissate per questioni di prestazioni/stabilità e compatibilità con hypervisor da loro usato.
2. **PaaS (Platform as a Service)**: ci viene fornito una scatoletta dove scrivere e eseguire/testare le applicazioni, il provider sarà il garante del fatto che il sistema sia abbastanza responsive e abbia un runtime sufficiente. E' meno flessibile rispetto allo IaaS ma allo stesso momento lo sviluppato perde meno tempo nella configurazione delle macchine.
3. **SaaS (Software as a Service)**: es. GMail quindi utente non può fare niente tranne usare il software.

In realtà il cloud vero è un po' più offuscato di quanto detto sopra, spesso non è facile collocare il modello di cloud di un'azienda esattamente in uno dei 3 modelli.
Un aspetto molto importante da considerare è quello economico, cioè grazie al modello pay-as-you-go cloud il costo capitale (CAPEX) si trasforma in costo operazionale (OPEX), inoltre il rischio di errore nella scegliere le macchine sparisce, se ho bisogno di più potenza richiedo una macchina più forte e se ho bisogno di meno potenza abbasso la potenza e risparmio. Allo

stesso momento i cloud service provider hanno diversi benefici: profitto sfruttando l'economia di scala (comprare un servo costa X, ma comprare N server non costa N*X), possono capitalizzare sui loro investimenti (amazon che vende la potenza residua) o possono sfruttarlo per promuovere un loro prodotto (es. per far usare .NET ai sviluppatori).

3.2.1 IaaS

IaaS fornisce automaticamente il processing, storage, network e altre risorse fondamentale per il computing, e l'utente può installare i software che sono per lui necessari.
Lo IaaS supporta queste 3 caratteristiche del cloud:

- Gestibilità e Interoperabilità: Attraverso un interfaccia può gestire tutte le VM che vuole, le può pagare come vuole;
- Disponibilità ed Affidabilità: gestita dal cloud provider attraverso le zone di disponibilità;
- Scalabilità ed elasticità: Proprietà del sistema ma deve essere implementato dall'utente.

L'architettura è composta dall'hardware seguita da un layer di virtualizzazione e l'utente ha accesso a due interfacce:

- Interfaccia per gestione delle risorse, queste risorse sono:
  - VM: creazione, cancellazione, gestione, sospensione delle VM
  - Virtual Storage: Allocare spazio, ridurre/aumentare del DB, scegliere servizi con velocità di lettura/scrittura diversi (per prezzo);
  - Virtual Network: gestione delle IP associate alle VM, registrazioni al dominio delle IP, scegliere la larghezza di banda.
- Interfaccia del il monitoring del sistema: viene messa a disposizione da VIM: L'orchestrazione delle macchine/risorse, in maniera rapida e dinamica, su un server viene gestita dalla Virtual Infrastructure Mangaser(VIM), diverse metriche vengono usate per il monitoraggio es:
  - VM: CPU usage, memory usage;
  - Virtual Storage: utilizzo del disco, livello di duplicazione dei dati e velocità di accesso al disco;
  - Virtual Network: L'uso della banda, lo stato di connettività e il bilanciamento sulla rete.

3.2.2 PaaS

PaaS mette a disposizione all'utente i linguaggi di programmazione e i tools (come IDE) supportati dal provider, quindi il controllo sul deployement dell'applicazione è in mani del consumer ma il controllo sulle infrastrutture sottostante è gestita dal cloud provider.
Alla base vi è una archittettura come quella di IaaS, vengono aggiunti i Runtime Environment e sopra questi ci sono i Programming IDE e le API/tools supportati dall'ambiente, spesso hanno in comune le funzioni come: computation e lo storage.
Il Runtime Environment si riferisce a una collezione di software, implementati con una collezione di librerie, le proprietà fornite dal Runtime Environment sono:

- Gestibilità ed Interoperabilità
- Performance ed Ottimizzazione
- Disponibilità ed Affidabilità
- Scalabilità ed Elasticità

L'interfaccia messa a disposizione per il controllo del sistema mette a disposizione un set di azioni

in base a:

- Policy based control: le azioni seguono delle regole per prendere delle decisioni quindi azioni seguite con if ¡¿ then ¡¿

- Controllo del workflow: descrizione del flusso di installazione e configurazione delle risorse oppure dei daemon.

3.2.3 SaaS

SaaS: vengono fornite al cliente applicazione del provider in cloud, l'utente non può gestire o sviluppare l'applicazione o gestire l'architetture nel cloud ma può interagire con esso attraverso interfacce ad es. portali/applicazioni web, che con introduzione del Web 2.0 ha potenziato l'iterabilità con l'applicazione in cloud, es: Facebook(Messenger), Office, Skype, DropBox, sistema CRM, sistema medico nazionale, sistema di trasporto pubblico.
Il punto fondamentale di questo tipo di cloud è l'Accessibilità e la portabilità.
Oss: La differenza tra il portale web e una pagina web è che il portale permette l'integrazione di diverse pagine attraverso dei login.

3.3 The 4 Cloud Deployement Models

I 4 modelli primari che differenziano un cloud, basati sulle possibilità di accesso, la dimensione e la proprietà dei servizi, sono:

- **Public Cloud** (o multi-tenant o external cloud): L'infrastruttura viene messa a disposizione del pubblico generale o almeno alle organizzazioni grandi attraverso pagamento per il suo uso, le caratteristiche sono:
  - Infrastruttura omogenea per garantire le medesime prestazioni a due utenti che pur trovandosi in posti diversi usano lo stesso servizio;
  - Policy comuni;
  - Risorse condivise;
  - Infrastrutture viene affittata;
  - Economia di Scala.
- **Private Cloud:** (o on-premise cloud o internal cloud) Viene operato da un'unica organizzazione e può essere gestita dalla stessa o un'altra organizzazione, cioè la categoria di utenti è limitata. A differenza del public cloud le caratteristiche sono:
  - Infrastruttura eterogenea per via del costo e dal fatto che una azienda privata non butta via un hardware solo per averli omogeneità;
  - Policy customizzate ad-hoc per gli utenti;
  - Risorse dedicate;
  - Infrastrutture gestita da house;
  - End-to-end control sui processi.
- **Community Cloud:** è una struttura gestita da diverse entità, in pratica più entità mettono insieme i loro cloud privati per generare un super cloud, ad esempio alcune università americane possono unire i loro cloud per qualche ricerca specifica ecc.
- **Hybrid Cloud:** è la comoposizioni di più tipi di cloud, ad esempio un azienda privata crea il suo cloud privato per la gestione di informazioni sensibili e usa il cloud pubblico per il resto, oppure lo usa per la scalabilità in caso di traffico aumentato.

4 Amazon ECOSYSTEM-IaaS

Amazon Web Service sono creati per lavorare indipendentemente ma possono interagire tra di loro, condividendo tra di loro le stesse conven-

zioni di nomi e autenticazione e minimizzando le connessioni interne.
Amazon mette a disposizione il concetto di regione e di zona di disponibilità:

- Ciascuna **Availibility Zone** è un data center indipendente con la propria griglia di potenza e connessione della rete, le zone all'interno di una regione sono collegate tra di loro attraverso connessione a latenza bassa. Il nome deriva dal fatto che sono zone ad alta disponibilità, quindi se una zona fallisce il suo effetto non viene notato dalle altre zone creando così una stabilità ed alta fault tolerance.
- **Regione** è un cluster di Availibility Zone localizzati in una area geografica. Quando si crea una istanza Amazon ci dà la possibilità di scegliere una availability zone (o di default lo si lascia far scegliere ad Amazon) e all'interno della stesso regione è possibile distribuire la propria istanza su più availability zone, quindi se una zona fallisce l'altra (molto probabilmente) continua a funzionare. Amazon non fa pagare i trasferimenti di dati all'interno della stessa regione, ma tra regioni le comunicazioni vengono fatte pagare.

Esiste AWS GovCloud che è una regione AWS isolata e permette di gestire dati estremamente sensibili e può essere acceduta solo da cittadini Americani verificate dal governo Americano e sul suolo Americano.
Uno dei servizi principali di AWS è **Simple Storage Service (S3)**, un storage object che è un sistema di archiviazione piatto cioè non è possibile creare cartelle al suo interno, i dati vengono salvati in formato binario e vengono distribuite automaticamente. L'accesso può essere effettuato da chiamate web (REST, Soap, BitTorrent) o query SQL e gli oggetti possono anche essere molto grandi (fino a 5TB ciascuno).
S3 sta alla base di tutta l'infrastruttura che sta alla base di Amazon e-commerce. Oltre a S3 Amazon fornisce anche un **EBS (Elastic Block Store)**, questo disco può essere formattato, montato e usato come hard disk locale, è un volume che persiste indipendentemente dal resto. Mentre le VM possono morire, i volumi rimangono sempre, e la garanzia di durabilità viene fornita da Amazon.
EBS è categorizzato in SSD e HDD, a seconda del tipo di macchine/dischi si paga in maniera diversa.
I **Security Group** definiscono l'insieme delle connessioni possibili per una certa istanza, e questi insiemi possono essere protocolli, porte, range delle IP.

4.1 EC2 - Elastic Cloud Computing

EC2(Elastic Cloud Computing) è uno dei servizi offerti per la creazione/gestione delle macchine virtuali on-demand. EC2 si basa sul concetto di data center programmabile, la possibilità di creare una propria infrastrutture (macchine in più availability zone o regioni) tramite linguaggi di programmazione. I concetti chiave che constituiscono EC2:

- **Amazon Machine Image** (AMI): è un file contenente una descrizione di una VM, cioè eventuali software e le configurazioni, possono essere pre-built, create, modificate e vendute. Ciascun AMI ha un ID unico;
- **Istanza**: rappresenta una copia di AMI in esecuzione, multiple copie di un unica AMI può essere messa in esecuzione;
- **Elastic IP address**: allocazione di un IP statico e collegarle alla propria istanza, ciascuna istanza può avere al più un IP statico.

I core messi a disposizione dal server (di cui non sappiamo praticamente nulla) vengono virtualizzati, dividendo prima in core poi dividendo ulte-

riormente il core-time. L'unità di misura è Elastic Compute units che corrisponde a Intel Xeon (o AMD Opteron) da 1.0-1.2 GHz del 2007.
Le risorse possono essere:

- Persistenti: anche in caso di fallimento del hardware Amazon ci garantisce la sua persistenza attraverso ridondanza, recupero automatico e failover automatizzato. Le componenti persistenti sono:
  - Elastic IP Address
  - EBS
  - Elastic Load Balancer
  - Security Groups
  - AMI slavate in S3 o EBS
- Effimere: l'utente deve garantire la sua ridondanza e mantenimento attraverso atri servizi di EC2, in caso di fallimento i dati salvati vengono in generale persi. Le istanze sono effimere.

I modelli del prezzo sono:

- **On demand**: Il pagamento avviene per le capacità (tempo) usate con nessun obbligo a lungo termine.
- **Reservati**: Si fa una sorta di abbonamento con un pagamento di una somma prima e poi le macchine si possono usare per la durata dell'abbonamento a un prezzo scontato, è disponibile in 3 tipi Light, Medium e Heavy e può essere per 1 o 3 anni e può far risparmiare fino al 71% rispetto a On-Demand. Attenzione non ci riserva delle risorse per la durata dell'abbonamento ma solo il prezzo.

- **Spot**: Si fa asta per la potenza computazionale non usata da Amazon EC2, sono quelle che costano di meno, perché non mi viene garantita la durata della macchina nel tempo, nel caso il prezzo offerto è maggiore del costo necessario per eseguire quella macchina Amazon darà la sua macchina, ma siccome il prezzo di elettricità è variabile e se dovesse superare il prezzo offerto Amazon può spegnere tale macchina senza avvisare.

Un altro indice di costo è **Data Transfer**, il trasferimento dei dati dentro o fuori dalla istanza EC2 vengono pagati (in base alla quantità) se non il trasferimento avviene al di fuori della regione. Il trasferimento dentro la regione è completamente gratuita.

4.2 AutoScaling

Uno servizio molto import di Amazon è **AutoScaling**(Fig 3). Per far funzionare AutoScaling è importante anche il **Elastic Load Balancer**, grazie al quale l'utente finale vedere solo un collegamento e Elastic Load Balancer pensa a distribuire le richiede alle nostre macchine. Le istanze EC2 possono essere categorizzate in auto scaling groups, di solito con un range tra un minimo e un massimo, se una nuova macchina entra in esecuzione oppure viene de-allocata il Load Balancer viene avvertito della modifica in

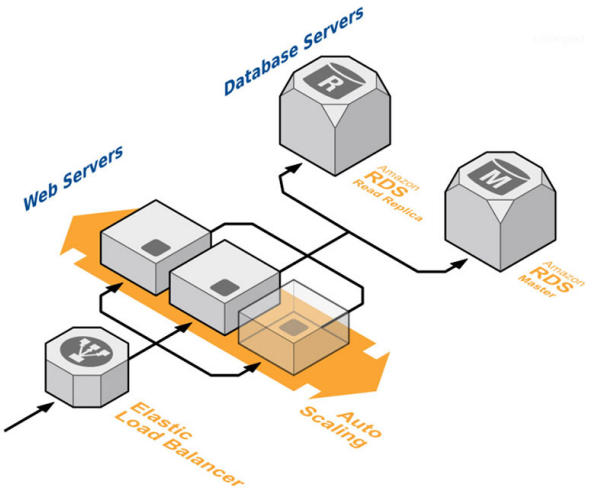

Figure 3: Esempio di Auto Scaling for AWS

modo che possa funzionare bene. Gli Auto Scaling group sono istanze EC2 che condividono caratteristiche simili e devono essere fatte per scalare, e aggiunge automaticamente un nuovo nodo in base a policy definite dall'utente, programmazione oppure health checks e workload e la nuova istanza può venire anche da availability zone diverse della stessa regione.
Auto Scaling group contiene solo un launch configuration che descrive le istanze che devono far partire (contiene i parametri delle AMI), se si aggiorna il launch configuration questo affliggerà solo le nuove istanze, le vecchie istanze rimangono tali ma esse vengono terminate prima durante un scale in.
Attenzione una macchina può essere o spenta o terminata, nel caso viene terminata viene eliminata completamente e AutoScaling group termina una macchina.
L'ecosistema di Auto Scaling è composta da:

- Cloud Watch: Per monitorare le istanze EC2, le metriche più usate per il CloudWatch sono:
  - CPU usage
  - Latenza
  - Numero di Richieste
  - Il numero di Host(macchine) in buona salute
  - Il numero di macchine non in buona salute
- Elastic Load Balancer: Per distribuire il lavoro su un numero qualunque di istanze EC2 ed effettuare controlli periodici sulla salute del nodo (basandosi sul tempo di risposta) e in caso non siano in buona salute il load balancer smette di inviare a lui le richieste.
- Auto Scaling: Utilizza i dati collezionati dal CloudWatch per costruire sistemi che posso scalare (dentro o fuori) all'intero del range

Il **Trigger** è un meccanismo di attivare una policy, in questo caso di aumentare o diminuire il numero dei nodi. Il trigger può essere attivato da un allarme cloud watch (configurato per guardare una metrica di cloud watch) oppure un auto scaling policy che descrive cosa fare in caso di un allarme. Quando si attiva il trigger lancia un processo chiamato Scaling activity che esegue la auto scaling policy. Osservazione Auto Scaling supporta ma non necessita Elastic Load Balancer. In generale almeno due trigger sono necessari, uno per Scale up e uno per Scale down, per mantenere un equilibrio desiderato. Un allarme è una metrica e verifica se questa metrica supera un limite stabilito per un certo tempo.
Un'allarme è un'entità in grado di osservare con continuità una certa metrica e ci indica se tale metrica ha superato un certo valore prefissato per un certo tempo, per creare un allarme è necessario specificare:

- Metrica da osservare
- Il threshold della metrica
- Il numero di periodo di valutazione

Gli stati in cui l'allarme si può trovare sono:

- Ok, tutto bene! Metrica è sotto il threshold.
- Alarm: Metrica è sopra il threshold, è necessaria un azione.
- Dati Insufficiente, metrica disponibile o non ci sono abbastanza dati.

Se l'allarme cambia lo stato e vi rimane per un determinato periodo di valutazione, un'azione viene invocata in base alla policy.
I scenari automatici dell'auto scaling sono:

- Fleet Management: assicura il performance ottimale della macchina, controllando la salute dell'istanza con Health Check: se un'istanza dovesse terminare questa viene individuata e si fa partire un'altra macchina.
- Scheduled scaling: Azioni vengono eseguite

in maniera programmata, quindi è una sorta di evento temporale per eventi ricorsivi o scheduled.
- Scaling Dinamico: In base all'allarme su una metrica c'è una politica che risponde all'allarme. L'azione può dipendere stepwise da gravità dell'allarme oppure facendo tracking di una certa metrica e aggiusto il sistema in modo da calmare l'allarme.

La terminazione della macchina può essere fatta anche per un ribilanciamento delle macchine nelle availability zone, la precedenza dello spegnimento viene data in modo da bilanciare tra le zone seguito dal fatto di voler preservare le macchine con il launch configuration più recente seguito dalla macchina prossima allo scatto dell'ora (per il prezzo che si paga ogni ora). Auto Scaling inizializza una nuova istanza prima di spegnerne una per non compromettere la disponibilità e le performance dell'applicazione. Dopo che è iniziata la fase di Auto Scaling esiste un periodo di cooldown, in questo periodo nessun altra attività di scaling può iniziare.
I canditati per autoscaling sono:

- Web Tier - sistemi che forniscono servizi web
- Application Tier
- Load Balancing Tier - sistemi che gestiscono il carico
- Stateless Tier - sistemi sono prive di stato es. funzioni pure, sono prive di memoria e quindi da un punto di visto funzionale non cambia se faccio autoscaling o no.

I candidati che non dovrebbero fare autoscaling:

- Database Relazionali
- Database non-relazionali
- Sistema di caching distribuito
- Elastic Search
- Sistemi con lo stato - es. Kafka, non avrebbe senso fare autoscaling automatico.

5 PaaS with Azure