Università Ca'Foscari Venezia
Facoltà di Scienze MM.FF.NN.

Corso di laurea in Scienze dell'Informazione

CLOUD COMPUTING

Relatore
Prof. Alessandro Roncato

Tesi di laurea di
Piergiorgio Matteazzi

Sommario

L'obiettivo della tesi è valutare il Cloud Computing dal punto di vista dei vantaggi/svantaggi che le aziende possono trarne, e quali motivi rallentano la diffusione di tale tecnologia.

- Capitolo 1 introduce il Cloud Computing a partire dalla sua definizione per arrivare a vantaggi/svantaggi che porta con sè.
- Capitolo 2 verranno presentati gli strumenti più usati per iplementare un Cloud Computing.
- Capitolo 3 viene riportato un esempio di linguaggio orientato ad ambienti Cloud.
- Capitolo 4 viene discusso un caso pratico per individuare eventuali vantaggi ad un approccio Cloud.
- Capitolo 5 Conclusioni.

Prefazione

Al giorno d'oggi gli utenti che fanno uso di un computer è cresciuto in maniera esponenziale, questo ha portato come risultato ad una maggior richiesta di soluzioni scalabili[1], ed un aumento considerevole dei dati scambiati tra gli utenti.

Inoltre la percezione che ha l'utente del calcolatore è cambiata di pari passo con la sua evoluzione: inizialmente erano macchine ingombranti costosissime per le quali si doveva attendere il proprio turno per usarle.

Successivamente l'utente associava al computer il terminale che gli permetteva di interagire con l'unità centrale. Con l'uscita del PC e di internet il terminale si è trasformato nel calcolatore vero e proprio che a sua volta con l'avvento del calcolo distribuito si è evoluto in un elemento del sistema di calcolo. Unendo i concetti di mainframe e di calcolo distribuito siamo giunti al Cloud Computing. Ne sentiamo parlare ormai ovunque e sembra quasi un'invenzione nata dal nulla, ma c'è chi parla di una strategia di marketing per il riutilizzo di tecnologie mature e già ampliamente utilizzate nel passato. La fusione tra le tecnologie di virtualizzazione ed il paradigma Service-Oriented Architecture ha permesso lo sviluppo di applicativi altamente portabili ad architettura distribuita, oltre alla possibilità di interconnettere ed utilizzare facilmente software eterogenei ed eventualmente gestiti da altre aziende.

Vi sono pareri discordanti riguardo Cloud Computing : Gartner Inc.[2] *sostiene che il Cloud Computing sia la maggiore evoluzione tecnologica offerta dalla rete Internet nonostante non sia ancora una tecnologia matura. Daryl Plummer*[3] *sostiene che il Cloud Computing creerà una nuova opportunità per plasmare il rapporto tra coloro che utilizzano i servizi IT e quelli che li vendono. In sostanza, ciò significa che gli utenti saranno in grado di concentrarsi su ciò che il servizio offre loro e non di come viene implementato*[38]. Dal canto suo Richard Stallman[4] sostiene che sia una trappola di marketing[2]: la sua tesi si basa sul fatto che sempre più utenti acquistano servizi in sistemi proprietari chiusi che costeranno sempre di più nel tempo. Aggiunge inoltre, che usando software proprietari si resta nelle mani dello sviluppatore; esorta quindi gli utenti a non affidarsi a sistemi altrui e a non condividere i pro-

[1] Il termine scalabilità, nelle telecomunicazioni, nell'ingegneria del software, in informatica e in altre discipline, si riferisce, in termini generali, alla capacità di un sistema di crescere o diminuire di scala in funzione delle necessità e delle disponibilità. Un sistema che gode di questa proprietà viene detto scalabile.

[2] Una società multinazionale leader mondiale nella consulenza strategica, ricerca e analisi nel campo dell'IT con oltre 60.000 clienti nel mondo.

[3] Vice presidente delegato Gartner.

[4] Padre del movimento per il software libero

pri dati in internet ma mantenerli in locale nel proprio pc. Alcune delle idee sostenute da Stalman, risultano essere anche le opinioni di molti consumatori. In uno studio pubblicato dalla direzione generale politiche interne della comunità europea proprio in merito al Cloud Computing[10], vengono esposti i risultati statistici, in cui il principale timore espresso dagli intervistati è quello sulla riservatezza dei dati aziendali che diventa quindi il principale freno potenziale per le PMI contattate, seguito dalla preocupazione per integrità dei servizi. La UE nel Settembre 2012 ha elaborato una strategia denominata *Sfruttare il potenziale del cloud computing in Europa* per cercare di ridurre la frizione del mercato nell'ottobre 2013 ha istituito una commissione costituita da un gruppo di esperti, incaricati di individuare clausole contrattuali sicure ed eque per i servizi di cloud computing.

Elenco delle figure

Indice

Capitolo 1

Cloud Computing Definizione

1.1 Introduzione

Nel seguente capitolo si cercherà di dare una definizione il più chiara possibile del Cloud Computing; successivamente verranno elencate le caratteristiche principali per poi definire quali sono i modelli ad oggi maggiormente riconosciuti; infine verrà data una valutazione vantaggi/svantaggi che questa nuova concezione porta con sè.

1.2 Definizione

Al momento non si è ancora focalizata una definizione de iure[1] del Cloud Computing, se poi consideriamo gli standard espliciti di riferimento c'è una continua evoluzione di nuove definizioni. L'unico punto chiaro si ha sugli gli standard delle tecnologie e dei protocolli del web, o più in generale di Internet. Vi è stata di conseguenza una proliferazione di definizioni che in molti casi convergono, anche se redatte a livelli diversi di astrazione, che ne danno una definizione qualitativa, e quindi sono fonte di confusione.

La pubblicazione *Break in the Clouds: Towards a Cloud Definition*[1] analizza 22 definizioni del Cloud Computing, ma ne esistono anche altre fatte da autorevoli analisti e organizzazioni. Di seguito viene riportata la definizione fornita dal NIST[2] :

Il cloud computing è un modello che consente l'accesso di rete diffuso, comodo e a richiesta a un insieme condiviso di risorse informatiche configurabili (per es. reti, server, archiviazione, applicazioni e servizi) che possono essere

[1] espressione con cui si indica la tendenza riformatrice

[2] National Institute of Standards and Technology

rapidamente messe a disposizione e rilasciate con uno impegno di gestione o un'interazione con il fornitore di servizi minimi.[11]

Nella definizioine data dal nist il modello cloud viene identificato da cinque caratteristiche essenziali, tre modelli di servizio e quattro modelli di distribuzione. Queste caratteristiche essenziali indicano come un servizio cloud deve essere strutturato, i modelli di servizio indicano cosa deve erogare e i modelli di distribuzione indicano da dove i servizi devono essere erogati (fig. 1.1).

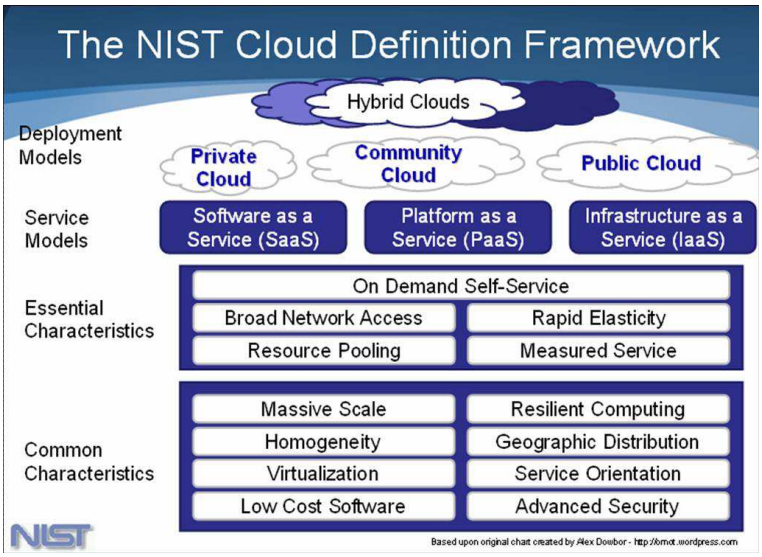

Figura 1.1: NIST cloud definition Framework

1.2.1 Cloud Computing caratteristiche essenziali

L'utente ha la facoltà unilaterale di approvvigionarsi di risorse computazionali (ad esempio tempo macchina e storage di rete) automaticamente, senza che ci sia la necessità di una interazione umana con i fornitori del servizio.

"Le risorse sono accessibili via rete attraverso standard che promuovano l'uso di piattaforme client eterogenee come, ad esempio smartphone, laptop,PDA[3], ecc.

Le risorse computazionali (storage, capacità elaborative, memoria, ecc.) del fornitore sono messe in comune per servire molteplici utenti, usando uno

[3]Personal Digital Assistant - si considerano PDA quei dispositivi portatili (palmari) con i quali è possibile gestire un'agenda, scrivere brevi messaggi, visualizzare o modificare documenti, riprodurre brani musicali (se si dispone di sufficiente spazio per immagazzinare i dati) ed in alcuni casi eseguire programmi e giochi (app).

schema multi-tenant[4], che gestisce risorse fisiche e virtuali, dinamicamente assegnate e riassegnate in accordo con le indicazioni degli utenti. Le risorse sono in grado di essere allocate rapidamente ed elasticamente, in alcuni casi in modo autonomo, per adattarsi in maniera veloce alle maggiori o minori richieste degli utenti. I sistemi ottimizzano l'utilizzo delle risorse tramite strumenti di misura, basati su adeguati livelli di astrazione. L'utilizzo delle risorse deve poter essere monitorato, controllato ed elaborato, sia dal fornitore che dall'utente del servizio. Citando l'ENISA[5] *il cloud computing è quindi, un nuovo modo di erogare servizi IT, non una nuova tecnologia.*

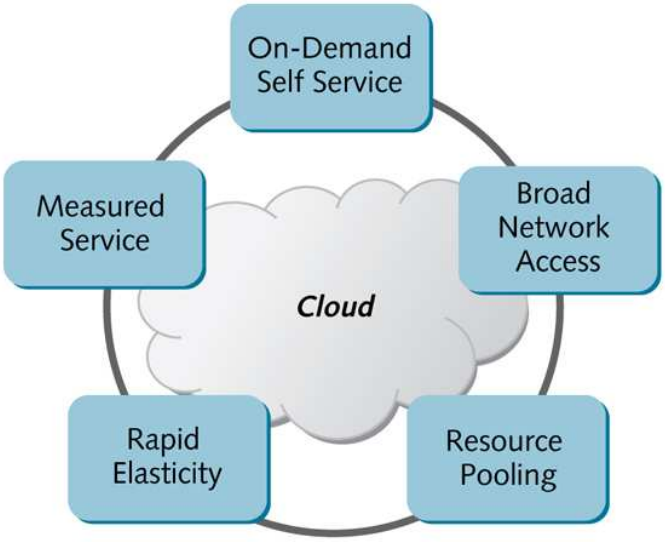

Figura 1.2: Caratteristiche essenziali

La flessibilità ed efficienza delle architetture cloud sono rese possibili da meccanismi oramai consolidati quali:

Astrazione: il Cloud Computing astrae i dettagli dell'implementazione ad utenti e sviluppatori. Le applicazioni girano su un sistema fisico che non viene specificato; i dati sono contenuti in luoghi non specificati; l'amministrazione del sistema è affidata ad altri e gli utenti accedono dovunque.

Virtualizzazione: il Cloud Computing virtualizza i sistemi attraverso pooling di risorse condivise. I sistemi e lo storage possono essere distribuiti da un'infrastruttura centralizzata. I costi sono stimati in base ai consumi; il multi-tenancy[6] è attivo e le risorse sono facilmente scalabili" [38].

[4] multi-cliente

[5] Agenzia Europea sulla Sicurezza Informatica

[6] La condivisione di risorse tra due o più clients

1.3 Modelli

In questo capitolo si definiscono le due diverse classi di Cloud che il NIST[7] ha introdotto, la prima basata sui modelli di erogazione l'altra su quelli di servizio.

Originariamente il modello non richiedeva alla nuvola di usare un pool di risorse che supportasse il multi-tenancy,[8] qualità invece richiesta nell'ultima definizione fornita dallo stesso organismo.

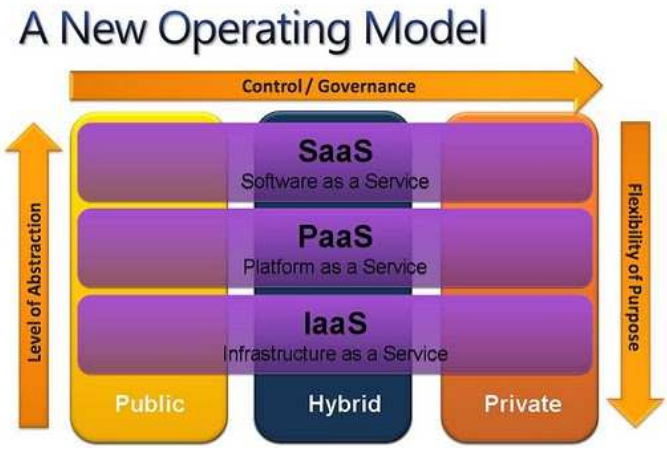

Figura 1.3: Classi cloud computing

1.3.1 Modelli di Servizio

Nei modelli di servizio le nuvole sono un'espressione del tipo di infrastruttura che viene erogata. Differenti vendors[9] offrono nuvole con differenti servizi associati ad essi. Tutti i modelli di servizio assumono la forma: XaaS (<Something> as a Service).

Il modello SPI[10] fig.1.4:

[7] Agenzia federale americana National Institute of Standards and Technology

[8] La multi-tenancy si riferisce ad un principio nell'architettura del software o dell'hardware in cui una singola istanza del software viene eseguita su un server, offrendo il proprio servizio a più client.

[9] In a supply chain, a vendor, or a supplier, is an enterprise that contributes goods or services.

[10] SPI is an acronym for the most common cloud computing service models, Software as a Service, Platform as a Service and Infrastructure as a Service.

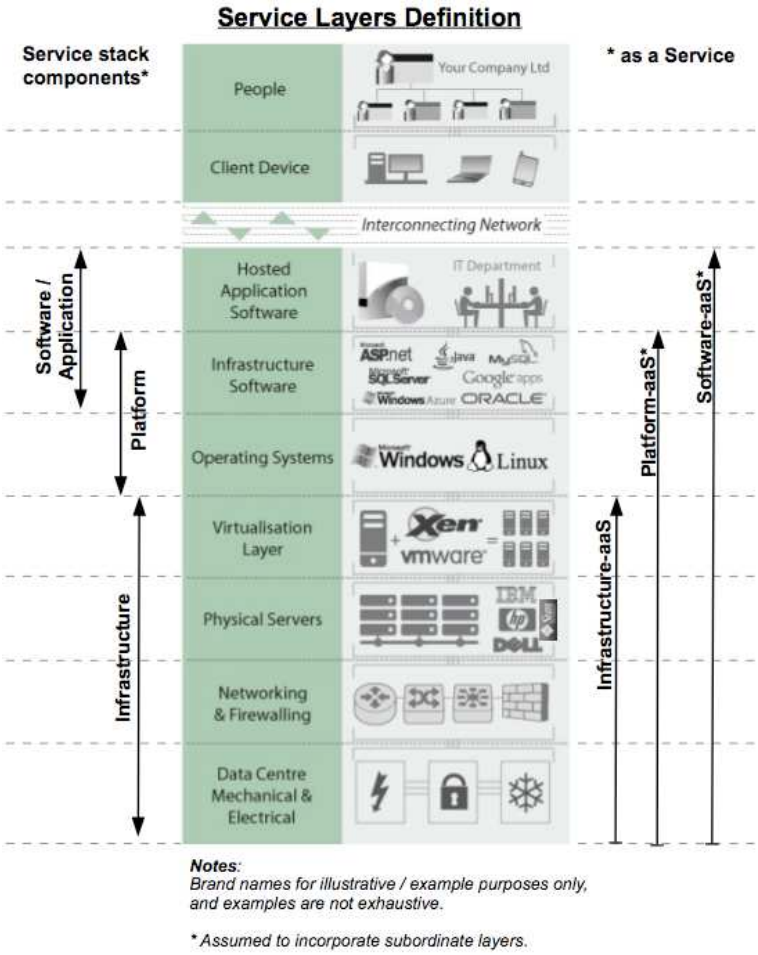

Figura 1.4: Modelli di servizio

Software as a Service: è un ambiente completamente operativo con applicazioni, gestione e interfaccia utente. Le applicazioni sono offerte al cliente attraverso una comoda interfaccia (di solito un browser) e la competenza dell'utente è limitata all'inserimento, alla gestione e all'interazione dei dati.

Tutto ciò che utilizza a partire dall'applicazione per arrivare all'infrastruttura è a carico del vendor. Tra i benefici di una soluzione di questo genere troviamo: una facile amministrazione, aggiornamenenti e un sistema di patch automatici forniti dal provider del servizio, una totale compatibilità data dalla stessa versione del software per tutti gli utenti del sistema e quindi una facile collaborazione, un'accessibilità globale fornita dal web.

"Platform as a Service: offre macchine virtuali, sistemi operativi, applicazioni, servizi, framework di sviluppo e strutture di controllo. Il cliente può

sviluppare le sue applicazioni usando l'infrastruttura cloud o le applicazioni programmate per mezzo di linguaggi e strumenti supportati dal provider.

Il provider gestisce l'infrastruttura cloud, i sistemi operativi e il software abilitante. Il cliente invece installa e gestisce le applicazioni che sviluppa. Troviamo molte caratteristiche interessanti: la possibilità di aggiornare e cambiare frequentemente le caratteristiche del sistema operativo.

Vari team di sviluppo software, distribuiti geograficamente, possono lavorare insieme ad un progetto. I servizi possono essere ottenuti da diverse fonti che attraversano i confini internazionali.

I costi iniziali e di mantenimento possono essere ridotti grazie all'utilizzo dei servizi di un infrastruttura gestita da un solo gestore rispetto all'uso di servizi offerti da diversi hardware che spesso soffrono di problemi di incompatibilità. Di contro si potrebbe incappare nel rischio di lock-in quando si usano servizi o linguaggi di sviluppo proprietari"[38].

"Infrastructure as a Service: mette a diposizione macchine virtuali, storage virtuale, infrastrutture virtuali e altre risorse hardware di cui i clienti possono usufruire.

Il provider che fornisce il servizio IaaS gestisce tutta l'infrastruttura mentre il cliente si occupa del suo sviluppo che include il sistema operativo, le applicazioni e le interazioni utente con il sistema.

Le sue caratteristiche includono: un servizio di tipo Utility computing, gestione automatica delle attività di tipo amministrativo, scalabilità dinamica, virtualizzazione dei desktop, connettività internet"[38].

Infrastructure as a Service

Esempi di gestori di servizi Infrastructure as a Service sono:

Amazon AWS (Amazon EC2)
Google Compute Engine
Windows Azure
IBM SmartCloud Enterprise
HP Enterprise Converged Infrastructure
Eucalyptus
GoGrid
FlexiScale
Linode
RackSpace Open Cloud
Terremark

L'esempio classico di Infrastruttura come Servizio è Amazon EC2. Provvede infatti, a fornire storage e macchine virtuali nelle quali vengono installati il sistema operativo e le applicazioni.

I clienti sono liberi di installare i sistemi operativi messi a disposizione attraverso immagini compresse oppure qualunque software vogliano.

La responsabilità di Amazon è espressa nello SLA[11] che lo obbliga a provvedere ad un livello di affidabilità commisurato al tipo di offerta richiesta, di solito espressa come tempo di attività del sistema senza interruzioni.

[11]Service Level Agreement (in italiano: Accordo sul livello del servizio) strumenti contrattuali attraverso i quali si definiscono le metriche di servizio (es. qualità di servizio) che devono essere rispettate da un fornitore di servizi (provider) nei confronti dei propri clienti/utenti.

Platform as a Service

Tra i provider che offrono Platform as a Service invece troviamo:

Google App Engine
Windows Azure Platform
IBM SmartCloud (Application Services)
Amazon Elastic Beanstalk
Caspio
GoGrid CloudCenter
Heroku
Mobile App Development Platforms

A fronte del pagamento di una tariffa adeguata, offrono il provisioning dinamico di un numero di nodi computazionali e di memorizzazione, adeguato a sostenere il livello di carico dell'applicazione.

Come detto sopra tra i più noti possiamo citare

Google AppEngine
Windows Azure
Amazon Elastic MapReduce

Ciascuna di queste vincola in modo più o meno specifico lo sviluppatore. La prima, per esempio, pur fornendo un runtime dedicato "al linguaggio Python, si appoggia sulle caratteristiche di virtualizzazione della Java JVM, sulle Java Persistence API e su Java Data Objects. Azure, al contrario, si appoggia su .NET, SQL server e SOAP. La platform in-cloud di Amazon, infine, si basa sul framework open source Hadoop, lo stesso che viene utilizzato da Yahoo!"[38].

Software as a Service

"Alcuni dei principali servizi che offrono cloud in Software as a Service sono:

GoogleApps
Oracle On Demand
Microsoft Office Live
SQL Azure
SalesForce.com (es., CRM)

Tra questi servizi spicca quello offerto da Google, offre infatti una suite di applicazioni pronte all'uso. Tra quali citiamo Gmail, Google Calendar, Google Docs, Google Sites che insieme alle altre permettono di rendere più efficiente la propria attività. A fronte di una spesa mensile offre, nella versione Business del pacchetto, oltre alla suite di applicazioni, uno storage di 25 GB ad utente, una disponibilità del servizio del 99,9% e un servizio clienti disponibile 24 ore su 24, 7 giorni su 7"[38].

1.3.2 Modelli di Distribuzione

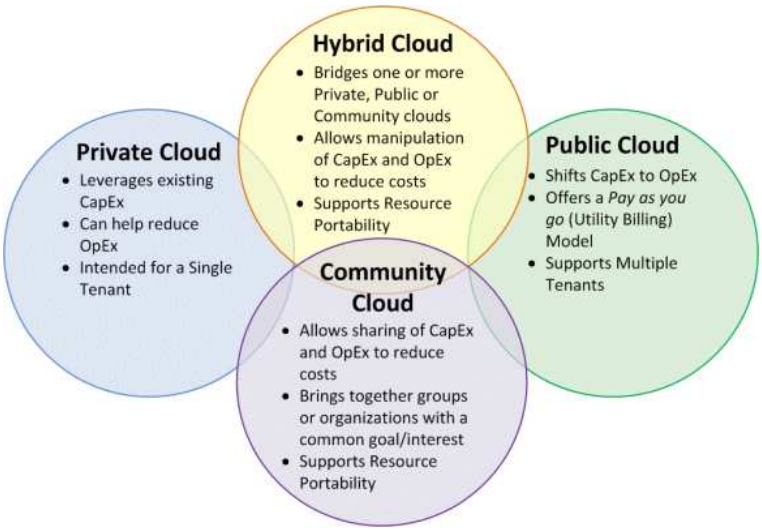

Figura 1.5: Modelli di distribuzione

I modelli di Distribuzione / implementazione, si dividono in:

Cloud Pubblica

Disponibile per un uso pubblico o per un grande gruppo industriale; è gestita da un provider che offre servizi cloud ed è accessibile attraverso internet. I servizi nella nuvola pubblica sono tipicamente offerti in maniera gratuita, con risorse limitate, oppure attraverso una soluzione a consumo. "Tra i vantaggi troviamo: una gestione facile, che non prevede un grosso esborso di denaro, in quanto hardware applicazioni e costi di banda sono a carico del provider; una scalabilità del sistema e una migliore gestione delle risorse date dal modello on-demand" [38].

Cloud Privata

Utilizzata esclusivamente da imprese. Potrebbe essere gestita dall'impresa stessa o data in gestione ad un'altra azienda, ed essere ubicata in sede oppure fuori sede. Chiamata nuvola interna o nuvola aziendale. Offre infatti servizi di hosting ad un numero limitato di utenti dietro ad un firewall. I progressi nella virtualizzazione e il calcolo distribuito hanno permesso agli amministratori della rete aziendale e dei data center di diventare effettivamente fornitori di servizi che soddisfano le esigenze dei loro clienti all'interno della società. "Concepita per venire incontro alle esigenze delle organizzazioni che vogliono un maggiore controllo sui propri dati rispetto a quello che possono ottenere utilizzando un servizio di terze parti come Amazon EC2 o Amazon S3 " [38].

Cloud Ibrida

"La nuvola ibrida combina nuvole private e pubbliche, dove ognuna mantiene la propria identità nonostante venga unita assieme alle altre. Può offrire un accesso ai dati e alle applicazioni di tipo standardizzato o proprietario.

Viene generalmente offerta con due soluzioni: un fornitore dispone di un cloud privato e costituisce una partnership con un provider che gestisce una cloud pubblica, o viceversa, un provider che gestisce cloud pubbliche costituisce una partnership con un fornitore che fornisce piattaforme su cloud private. Una Cloud Ibrida è un ambiente di cloud computing in cui l'organizzazione fornisce e gestisce alcune risorse in-house mentre altre vengono fornite esternamente. Ad esempio, un'organizzazione potrebbe utilizzare un servizio di cloud pubblico, come Amazon S3 mantenendo in-house i dati di archiviazione dei clienti.

Idealmente, l'approccio ibrido consente alle aziende di trarre vantaggio dalla scalabilità e dal rapporto costo-beneficio che un'ambiente di cloud computing pubblico offre, senza esporre applicazioni critiche e dati a vulnerabilità di terze parti" [38].

Cloud Comunitaria

"Nuvola comunitaria è una nuvola che è stata pensata per offrire funzionalità a scopi comuni. Una nuvola privata, quindi, differisce da una pubblica, non solo per il fatto che viene usata abitualmente sfruttando reti interne e quindi garantisce un maggior livello di sicurezza, ma anche per la dimensione poichè si utilizzano da un centinaio ad un migliaio di nodi, mentre in quella pubblica i nodi sono anche decine di migliaia.

Si contrappone un approccio single-tenancy della nuvola privata a quello multitenancy di quella pubblica, e il modello di pricing si basa sul concetto di capacità utilizzata e non su quello di uso.

La nuvola ibrida, come visto, include caratteristiche di entrambe: un'azienda potrebbe decidere di mantenere nella nuvola privata applicazioni del core business e dati sensibili, mentre in quella pubblica i servizi non core" [38].

1.4 Considerazioni

1.4.1 Modelli di servizio

Per poter meglio valutare il Cloud Conputing è importante capire le modalità con cui un'organizzazione si può avvalere dei servizi dell'Information Technology, come mezzo attraverso il quale condurre e migliorare il proprio business.

"Molte organizzazioni, o enterprise[12], possiedono e gestiscono in proprio un complesso tecnologico, detto anche data center, che contiene l'infrastruttura informatica, dove sono mantenuti i dati e sono eseguiti i servizi utili per il business e applicazioni, sviluppate internamente, o da fornitori esterni, sono di proprietà dell'organizzazione così come le licenze d'uso delle applicazioni acquistate dai software vendor.

In questo modello, l'organizzazione si deve far carico delle spese dei capitali di investimento, necessari per acquistare sia i sistemi informatici e di comunicazione (server, sistemi di storage, router, ...), sia i sistemi che ne garantiscono il corretto e sicuro funzionamento (gruppi elettrogeni, impianti di condizionamento, sistemi di sicurezza, ...).

Oltre a ciò bisogna aggiungere le spese operative per l'esercizio e la gestione di tutta l'infrastruttura presente nei data center e quelle per il personale che gestisce i sistemi e le applicazioni. Allo scopo di ridurre i costi dell'IT, le organizzazioni hanno fatto sempre più spesso uso di organizzazioni esterne specializzate, dette ISP (Internet Service Provider), che nel tempo si sono evoluti (da ISP 1.0 a ISP 5.0) dando sempre più valore aggiunto alle organizzazioni che se ne servono"[37].

Nella loro evoluzione gli ISP sono passati da fornire servizi di connettività alla rete (prima generazione), al fornire anche servizi di utilità come la posta

[12]Una Enterprise è un insieme di organizzazioni con un insieme di obiettivi e di profitti comuni, ad es. un'agenzia governativa, un'impresa globale, un insieme di organizzazioni distanti geograficamente ma legate da una proprietà comune.

elettronica o le registrazioni a nomi di dominio DNS[13] (seconda generazione). I modelli di fornitura di servizi da parte degli ISP di terza, quarta e quinta generazione sono stati denominati rispettivamente Colocation, Application Service Provider (ASP) e Cloud Computing.

Modello Colocation

"La Colocation (o anche housing) è la concessione in locazione a un'organizzazione cliente, da parte di un ISP di terza generazione, di uno spazio fisico, generalmente all'interno di appositi armadi, detti rack, dove sono posti i server di proprietà di un'organizzazione cliente. Tipicamente i server sono ospitati in data center, in cui il Service Provider fornisce e garantisce la gestione degli aspetti infrastrutturali (energia elettrica, connessioni di rete, condizionamento, sicurezza, ...).

I server e le applicazioni sono invece gestiti da remoto direttamente dall'organizzazione cliente tramite Internet. In questo caso, l'organizzazione cliente si fa carico delle spese di investimento dei server e delle licenze di uso del software, ma non di quelle relative agli aspetti infrastrutturali necessari"[37].

Modello ASP

"Con il modello ASP il Service Provider fornisce alle organizzazioni clienti l'infrastruttura IT, ma soprattutto le applicazioni fornite come servizio attraverso la rete Internet. Il Provider possiede, gestisce e garantisce il corretto funzionamento dei server e del software, in base a specifici livelli di servizio concordati con le organizzazioni clienti. Le applicazioni e i servizi sono usati attraverso web browser (ad esempio Firefox, Internet Explorer, Chrome, Opera, ...), oppure con specifiche applicazioni client fornite dal Provider.

I servizi, quando non sono forniti gratis, sono fatturati a consumo, oppure secondo dei canoni periodici. Caratteristica tipica del modello ASP è la fornitura ad ogni organizzazione cliente di uno o più server dedicati, sui quali vengono eseguiti applicazioni e servizi, anch'essi dedicati, in modalità multi-istanza[14], secondo un modello Single-Tenancy, in cui ogni cliente ha la sua istanza dedicata dell'applicazione, spesso anche personalizzata ad hoc.

[13] Domain Name Server è un sistema utilizzato per la risoluzione di nomi dei nodi della rete (in inglese host) in indirizzi IP e viceversa.

[14] Multi-tenant si riferisce ad una architettura software in cui una singola istanza del suddetto software gira su un server ed è utilizzata da più di una client organization (tenant). La multi-tenancy rappresenta il concetto opposto all'architettura multi-istanza, nella quale separate istanze del software sono dedicate alle client organization.

Con questo modello l'organizzazione cliente non ha più spese legate agli investimenti hardware e software"[37], ma sostiene solo le spese operative legate all'uso dell'infrastruttura e del software. Il modello ASP ha introdotto il concetto di fornitura di software come servizio (SaaS), pagato in modalità flat o anche a consumo.

Modello Cloud Computing

Questo nuovo paradigma di elaborazione è dinamicamente scalabile[15] e fornisce come servizio qualunque capability[16] richiesta all'IT. "Il Cloud Computing consente l'accesso on-demand a risorse condivise che risiedono in data center massivamente scalabili, cui si può accedere in modo ubiquo da qualunque dispositivo connesso a Internet.

La prerogativa principale del nuovo modello è quella di trasformare l'IT da un centro di costo interno, in un insieme di servizi esterni, agili reattivi e pagati a consumo, da usare non solo come strumenti di business, ma come mezzo attraverso il quale condurre il business.

Analogamente al modello ASP, il Cloud Service Provider si fa carico dell'infrastruttura IT e dei servizi che fornisce alle organizzazioni clienti, le quali hanno solo l'onere di pagare ciò che consumano"[37]. L'organizzazione cliente, invece, ha maggiori vantaggi, perché la condivisione delle risorse consente l'abbattimento significativo delle spese operative.

Riassumendo quanto detto fino ad ora in merito alle varie possibilità che un azienda ha di avvalersi di risorse informatiche:

- L'azienda con datacenter di sua proprietà ha il costo più alto di gestione.

- Con il colocation l'organizzazione cliente risparmia, continuando però a sostenere spese sostanziali per l'infrastruttura informatica e per la sua gestione.

- Con il modello ASP l'organizzazione cliente ha un sensibile risparmio, perché non sostiene più gli investimenti per l'acquisto dell'infrastruttura informatica, ma solo le spese operative di uso delle applicazioni.

- Il Cloud Computing abbassa anche le spese operative, perché il modello favorisce la condivisione delle stesse risorse fra più organizzazioni clienti e riduce l'onere di configurazione dei servizi da parte del Service Provider, essendo una parte demandata direttamente alle organizzazioni clienti.

[15] La scalabilità è la capacità di un sistema di adattarsi all'aumento di carico (ad esempio elaborativo o di storage,...) aggiungendo più risorse in funzione delle necessità.

[16] Per capability, in questo contesto, si intende, in un'accezione generale, qualunque funzionalità, risorsa o azione fornita dall'IT.

Inoltre come visto in precedenza nella definizione del NIST, il cloud computing è in grado o promette di offrire diverse caratteristiche che risultano utili per gli utenti finali e in particolare per le aziende.

1.5 Vantaggi

Il passaggio ad una soluzione Cloud introduce i seguenti vantaggi:

La gestione dell'infrastruttura è messa a disposizione dal Cloud Provider.

Sicurezza e affidabilità dei Data Center, in quanto i Cloud Provider attuano una politica di sicurezza e sorveglianza nei propri Data Center.

Permette di passare da CAPEX[17] a OPEX[18], il pagamento avviene in base all'utilizzo. Questo permette un notevole risparmio soprattutto nel caso di un ambiente di test e sviluppo, dove le macchine servono per un periodo di tempo limitato e rimarrebbero poi inutilizzate, dando la possibilità di reinvestire i capitali in altre attività produttive.

Focalizzazione sul proprio core busines: vi è una minor necessità di assumere personale specializzato che si occupi della gestione dell'infrastruttura.

Piani di disaster recovery: in molti casi le aziende che forniscono la soluzione cloud prevedono la possibilità di mettersi al sicuro da problemi che potrebbero accadere al Data Center.

Alta scalabilità: il cloud computing permette di reagire rapidamente a picchi improvvisi, per i quali non si potrebbe fare nulla in un ambiente tradizionale. Lo scaling è anche possibile verso il basso, per cui possiamo rilasciare altrettanto rapidamente risorse computazionali e storage non più necessari.

Facilità e autonomia di utilizzo: la larga maggioranza dei servizi cloud sono forniti con un interfaccia Web che permette anche agli utenti meno esperti la creazione di nuove istanze e il monitoraggio del funzionamento; la loro funzionalità inoltre è simile fra i vari provider, fornendo così una visione omogenea che facilità l'utente finale.

Riduzione dei rischi di impresa e dei costi operativi e di manutenzione: come per i servizi di outsourcing, modello al quale spesso il cloud si avvicina molto, l'azienda, delegando i suoi sistemi a una entità esterna specializzata in grado di fornire un servizio a costi inferiori, può ridurre i propri costi; si facilita inoltre una riorganizzazione interna, consentendo lo spostamento

[17] CAPital EXpenditure, ovvero spese per capitale, si intendono quei fondi che una impresa impiega per acquistare asset durevoli, ad esempio macchinari.

[18] OPerating EXpenditure, ovvero spesa operativa, è il costo necessario per gestire un prodotto, un business od un sistema altrimenti detti costi di O&M (Operation and Maintenance) ovvero costi operativi e di gestione.

di esperti IT dai compiti di più basso livello quale la riparazione di server aziendali, verso attività di innovazione e ottimizzazione delle soluzioni esistenti.

1.6 Svantaggi

Il cloud computing è una modalità recente che deve ancora arrivare ad una piena maturazione; per questo esistono ancora delle problematiche aperte che vengono via via affrontate nell'ambito della ricerca.

Il cloud computing, per la sua stessa natura di servizio usufruibile attraverso una rete accessibile da chiunque in qualunque posto, è nato grazie alla rete Internet e dipende dalla capacità di Internet di offrire una connessione efficiente e affidabile; un problema di connettività può provocare un blocco completo del processo produttivo legato alla disponibilità dei sistemi informativi cloud.Queste problematiche possono essere mitigate da un service provider: tramite soluzioni applicative o middleware (Google, per esempio, fornisce degli strumenti per poter lavorare offline sull'email e sugli altri strumenti offerti) tramite la fornitura di un servizio di rete sottoposto a un Service Level Agreement (SLA), che può essere rispettato sfruttando una connettività dedicata per l'accesso oppure basandosi su una connessione Internet con uno SLA specificato.

In Italia la mancanza di una banda larga e affidabile, distribuita su tutto il territorio, rende spesso difficile per molte aziende localizzate al di fuori dei grandi centri urbani l'accesso a servizi di tipo cloud; dove invece è presente, il limite è spesso nelle velocità ridotte di upload che non permettono grossi trasferimenti di dati verso il cloud, in particolare nelle fasi iniziali di migrazione. La soluzione è la posa di reti in fibra ottica, che si sta lentamente estendendo anche al di fuori dei centri maggiori. Meno grave, invece, è il problema del trasferimento tra istanze nel cloud, anche perché da parte dei provider è stato fatto un grosso lavoro sui protocolli e sulle strategie di implementazione.

Pericolo di perdita di controllo, nel passaggio al Cloud diversi enti/uffici aziendali potrebbero farlo senza appoggiarsi all'IT aziendale generando quindi comportamenti a rischio per la sicurezza per l'utilizzo di software non verificato/autorizzato, per esposizione dei dati sensibilio al Cloud.

Perdita di conformità: Impossibilità di mantenere certificazioni acquisite. Indisponibilità del provider per audit di terze parti, indisponibilità del provider ad acquisire certificazioni. Mancanza di requisiti per specifiche certificazioni (es. PCI DSS[19])

[19](Payment Card Industry Data Security Standard) Lo standard PCI DSS è costituito

La cancellazione dei dati non implica l'immediata cancellazione fisica e sopratutto sicura (con algoritmi certificati) degli stessi all'interno dell'infrastruttura cloud.

Non sempre il passaggio da CAPEX a OPEX porta vantagi, ci sono diversi costi nascosti soprattutto per la migrazione di servizio verso il Cloud.

Scarsa preparazione a gestire il modello Cloud, ad esempio se si lasciano accese le macchine virtuali quando non sono usate alzerà significativamente i costi di esercizio.

Attenzione alle responsabilità dirette e indirette! Controllare sempre bene i contratti verso i Cloud Provider.

Integrazione con le periferiche locali (Come stampanti o altri dispositivi USB) e con gli apparati di sicurezza è complessa (ed in alcuni casi impossibile).

Va comunque ricordato come il cloud non sia una soluzione per tutti i workload[20]: non conviene in questo momento migrare quelli business critical, che richiedono livelli molto alti di disponibilità e basse latenze di accesso, anche se futuri miglioramenti delle connessioni potranno ridurre queste limitazioni.

Privacy e sicurezza dei dati

Una delle prime domande degli utenti riguarda la sicurezza dei dati e delle informazioni caricate sul cloud. Ciò è strettamente legato anche alla convinzione che i dati conservati in un server sotto il controllo dell'azienda siano più sicuri e meno esposti a rischi; anche se questa convizione non è del tutto sbagliata, bisogna comunque constatare come i provider più importanti del settore cloud siano dotati di datacenter che adottano politiche per la sicurezza molto stringenti e anche le soluzioni sviluppate per cloud privati prevedono, fin dall'inizio, una parte dedicata alla sicurezza, con considerazioni sulla replicazione dei dati e la high availability.

Per quanto riguarda la privacy, la normativa del settore è abbastanza lacunosa e non in linea rispetto alle evoluzioni tecnologiche, per cui il giudizio spesso può dipendere dalle singole interpretazioni. La legge vigente è comunque quella del paese dove è localizzato il datacenter e solo alcuni provider permettono di scegliere quello da utilizzare: questo permette di poter valutare

da un elenco di requisiti di protezione in termini di gestione, criteri, procedure, architettura di rete, progettazione di software e altre misure di tutela dei dati sui titolari di carta. I principi fondamentali, per essere a norma sono: creare e gestire una rete protetta, proteggere i dati sui titolari di carta, adottare un programma di gestione delle vulnerabilità, monitorare e testare regolarmente le reti, adottare criteri di protezione delle informazioni.

[20] Carico di lavoro

dove ci siano maggiori garanzie e un'azienda italiana, per esempio, potrà scegliere un datacenter in Germania consapevole che i dati memorizzati in tale paese saranno protetti dalla legislazione europea.

Rimane comunque il problema in alcuni settori, quello sanitario ad esempio, nel quali la legislazione vigente sconsiglia o impedisce il trasferimento di dati al di fuori dei confini italiani. In questo caso e, in genere, in tutti i settori nei quali le esigenze di privacy e sicurezza sono molto elevati, si preferisce orientarsi verso scenari di hybrid o private cloud, che pur avendo costi iniziali maggiori, permettono poi di godere di tutti i vantaggi offerti dal cloud computing.

Vendor lock-in

Uno dei problemi più importanti dal punto di vista degli utenti è quello del cosiddetto lock-in[21]: infatti, se la migrazione sul cloud richiede grosse modifiche alle applicazioni esistenti, diventa poi anti economico il passaggio a un altro provider di soluzioni cloud: al momento questo problema è più sentito man mano che si sale di livello come soluzioni.

A livello di infrastruttura, infatti, viene in genere fornita una macchina virtuale che può essere facilmente migrata da un provider all'altro ed esistono anche formati standard come l'Open Virtualization Format (OVF). Per quanto riguarda la gestione, non c'è tuttavia una API standard: quella più usata è quella proposta da Amazon.

A livello di Platform e Software, invece, la maggior parte dei provider, pur sfruttando a basso livello i tipici protocolli di comunicazione di rete, espone all'utente soluzioni proprietarie; decidendo, per esempio, di adottare le API di Google App Engine, ci leghiamo strettamente a questa piattaforma, con il rischio di dover sottostare a cambiamenti arbitrari di prezzo e di doverci fidare delle garanzie degli specifici provider.

La soluzione è quella di andare verso standard aperti, con specifiche libere sulle componenti base e comuni a tutte le soluzioni cloud, mentre la concor-

[21] Il fenomeno del "lock-in si ha quando, individualmente o collettivamente, si è catturati da una scelta tecnologica potenzialmente inferiore rispetto ad altre disponibili, è assai rilevante nell'ambito delle tecnologie di Internet. Consideriamo due motivi che possono generare la cattura da parte di una tecnologia. Sovente vi sono dei costi fissi non recuperabili (o sunk costs), che rendono sconveniente cambiare tecnologia. Per esempio, il tempo per l'apprendimento di un certo software rappresenta un tale costo fisso. Il secondo è strettamente legato alla presenza di esternalità di rete. In questo caso, vi è un problema di coordinamento tra gli utenti di una certa tecnologia. Per esempio, come nel fenomeno del technology skipping, si può decidere di non passare a una nuova tecnologia superiore perché quella vecchia, avendo una maggiore base installata, genera maggiori benefici dovuti alle esternalità di rete"[37].

renza avviene sulla qualità dell'implementazione e su funzionalità avanzate ed innovative; una delle proposte più interessanti è quella di OpenStack[14], che propone delle API a livello IaaS rilasciate con licenza libera da utilizzare al posto delle API Amazon, attuale standard de facto.

Un'altra alternativa è di utilizzare una libreria che astragga dalle particolarità delle singole piattaforme di virtualizzazione, come per esempio libvirt3[15] , rilasciata sotto una licenza di tipo open source.

Un altro problema del vendor lock-in è quello dei dati: è inutile infatti avere delle API standard se poi non riusciamo a migrare dati esistenti da un provider all'altro. Per ovviare a questo problema, Borenstein e Blake[16] sostengono la necessità di introdurre standard valutativi del cloud, che, similmente agli standard ISO 9000, valutino l'affidabilità di un'azienda e la sicurezza che essa garantisce.

Gestione delle risorse

Anche la gestione dinamica delle risorse rimane un problema ancora molto complesso, dal punto di vista sia del datacenter sia dell'applicazione finale.

Se infatti come si è detto poco fa, uno dei vantaggi del cloud per l'utente del servizio è quello della visione dell'hardware potenzialmente infinito, non va dimenticato come a livello fisico ci sia in ogni caso dell'hardware, che va scalato e suddiviso tra i vari utenti mediante tecniche di prenotazione delle risorse future, e di monitoraggio delle risorse esistenti.

Particolarmente difficoltosa è la gestione dello storage, che deve poter offrire una grande scalabilità e elasticità (spesso richiedendo la creazione di nuovi algoritmi e sistemi di controllo), senza però perdere di vista la protezione dei dati conservati. Lo stesso dimensionamento complessivo di un datacenter è un problema non indifferente, poichè va garantito un buon compromesso tra numero di utenti servibili e investimenti necessari, supportando la flessibilità richiesta dal paradigma cloud.

Mentre dal lato applicativo soluzioni di tipo Platform as a Service riescono a fornire uno scaling dinamico dal punto di vista sia computazionale che dello storage, a livelli inferiori i problemi sono più complessi e, nel caso del livello Infrastructure, si deve comunque operare a livello di istanze che vanno lanciate ed eliminate al bisogno; se questa operazione risulta troppo complessa, l'utente potrebbe essere tentato di lasciare il sistema sempre avviato, riducendo uno dei possibili vantaggi offerti.

Prendendo in considerazione il lato dell'utente, va sottolineato come i provider garantiscano solo dei parametri indicativi per quanto riguarda CPU, I/O e connessione; nei momenti di maggior carico e in base alle ottimizzazioni dei vari hypervisor, le prestazioni a livello di una singola macchina

virtuale potrebbero ridursi, e quindi è necessario tenere in considerazione anche questo fattore, cioè se le applicazioni sul cloud hanno dei requisiti per questi elementi.

Licenze dei software

Le licenze attuali dei software commerciali hanno molto spesso delle restrizioni sul numero di computer sui quali esso può essere in esecuzione; questo ovviamente pone dei problemi in un ambiente cloud, dove si applicano modelli di pagamento diversi e diventa difficile valutare il numero di computer o di utenti.

Per questo motivo, i software sono principalmente resi disponibili con due tipi di licenze in ambiente cloud: pay as you go (PAYG), in cui si paga in base al numero di ore di utilizzo oppure bring your own licence (BYOL), nel quale è l'utente a doversi preoccupare di disporre di licenze compatibili con l'uso in cloud.

Il primo modello è ovviamente quello più moderno e facilita l'utente, ma al momento dipende in larga base da accordi specifici tra cloud provider e produttori di software: per fare un esempio, Microsoft SQL Server è disponibile con una licenza PAYG su Amazon EC2, mentre su IBM Smart Cloud questo modello di licensing è offerto principalmente per software di IBM stessa e per pochi altri software di produttori indipendenti.

Per quanto riguarda il secondo modello, nascono diversi problemi che sono emersi con lo sviluppo della virtualizzazione, tecnologia come detto alla base del cloud: in questo caso non abbiamo infatti una visione diretta dell'hardware sottostante e il modello di pagamento più comune, quello in base al numero delle CPU (o dei socket), perde chiaramente di senso e rischia di ridurre i vantaggi offerti dalla nuova tecnologia.

Dal lato degli sviluppatori di software il nuovo paradigma pone nuove problematiche di licensing sia dal punto di vista strettamente economico che tecnico; quest'ultimo problema è stato affrontato nel corso di diversi studi, fra i quali già nel 2009 *License management for grid and cloud computing environments.*[7] nel quale si propone GenLM, un sistema di management che sfrutta la crittografia asimmetrica, per consentire al provider di autorizzare e fatturare il servizio a livello di singola operazione, situazione piuttosto comune in un ambiente di grid e cloud computing. Ciò avviene calcolando l'hash dei dati in input e firmando poi tale hash, che viene quindi rimandato all'utilizzatore per poter essere mandato in esecuzione.

Capitolo 2

Cloud Computing

2.1 Introduzione

Fino a questo punto abbiamo esposto i concetti basilari che definiscono il Cloud Computing. Per poter meglio comprendere quanto sopra descritto, per poter avere gli strumenti di valutazione dell'adeguatezza di una scelta verso una soluzione Cloud, si ritiene utile conoscere quali sono gli strumenti software a disposizione dei provider che erogare servizi Cloud in modo sicuro e scalabile. Una struttura che eroga servizi Cloud necessita di un software di virtualizzazione e di un infrastruttura di rete, come esposto nell'articolo Cloud Computing : A preliminary look. In Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services.[35]

2.2 Virtualizzazione

Una delle tecnologie presente in molti degli aspetti che riguardano il cloud è proprio la virtualizzazione; il fatto di interagire con risorse reali in maniera trasparente astraendole attraverso una rappresentazione virtuale, ne permette un utilizzo più efficente ed equo. La virtuallizzazione svolge lo stesso compito che un sistema operativo compie per astrarre le risorse ai processi, ma tale astrazione è rivolta verso l'utilizzatore. Con le attuali tecnologie di virtualizzazione possono essere virtualizzate risorse hardware, come CPU, memoria RAM, memoria di massa o interi computer, o risorse software, come i sistemi operativi. Le tecnologie di virtualizzazione si basano su due componenti fondamentali:

- **Virtual Machine** è la rappresentazione virtuale di un intero calcolatore, dotata di tutto l'hardware che si ritiene necessario, e funziona

da contenitore logico del sistema operativo ospite; può essere memorizzata come immagine del disco rigido del computer, più alcune meta informazioni, come le risorse disponibili e le loro caratteristiche; è interessante notare che una macchina virtuale può essere spostata da un server a un altro, cosa molto importante nel contesto del Cloud Computing.

- **Hypervisor** chiamato anche Virtual Machine Manager (VMM), è il componente che gestisce i sistemi operativi ospiti in esecuzione su un server fisico, e presenta loro una vista virtualizzata delle risorse hardware fisiche.

2.2.1 Hypervisor

Gli Hypervisor si classificano in nativi, chiamati anche bare-metal hypervisor, o ospiti (fig. 2.1).

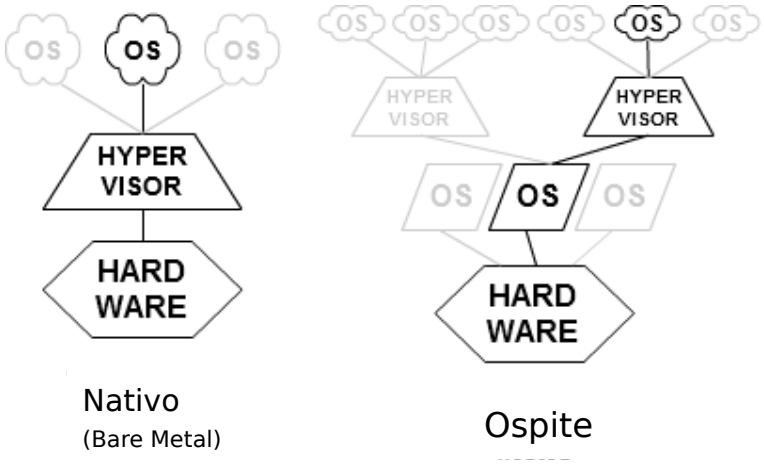

Figura 2.1: Classificazione Hypervisor

Hypervisor nativi

Questa tecnologia fa uso di software eseguito direttamente sull'hardware, che ne permette la virtualizzazione; tale immagine virtuale viene a sua volta gestita da un ulteriore sistema operativo che ne astrae le risorse ai processi. Per questo motivo, sono a volte chiamati hypervisor bare metal. Esempi di Hypervisor nativi sono ESXi di VMware, Hyper-V di Microsoft o Xen Hypervisor, un progetto open source usato da molti Cloud provider, come Amazon e Rackspace.

Hypervisor ospiti

In questo caso l'Hypervisor è visto come un processo del sistema operativo, che è a sua volta in esecuzione sull'hardware fisico. Risultano chiaramente meno performanti di quelli nativi, ed hanno, in generale, scopi diversi. La loro principale utilità è legata al fatto che permettono di utilizzare contemporaneamente sia il sistema operativo ospite che quello ospitante. Esempi di Hypervisor ospiti sono i vari prodotti di VMware per sistemi desktop, Oracle VirtualBox, Windows Virtual PC, Parallels Workstation, CoreOS, WLPAR, Solaris Zones.

Tipi di virtualizzazione

La virtualizzazione può essere supportata in modi diversi, i vari metodi si distinguono per il livello di astrazione e le performance fornite.

2.2.2 Virtualizzazione completa

La tecnologia di questo tipo fa uso di una virtualizzazione ed astrazione totale dell'hardware; il sistema operativo ospitato non necessita di modifiche software, questo perchè tutte "le sue chiamate di sistema vengono intercettate dall'Hypervisor ed emulate con istruzioni speciali. Questo approccio è il più generico e flessibile, poiché separa completamente il sistema operativo ospitato dall'hardware sottostante, tuttavia porta ad un sovraccarico computazionale che influisce negativamente sulle performance"[39].

2.2.3 Para Virtualizazione

Questo approccio richiede invece che il "sistema di virtualizzazione" renda disponibili ad ogni macchina virtuale delle interfacce hardware, tramite un sistema di astrazione software che permette l'accesso a tali interfacce in modo simile, ma non identico. La conseguenza di tale approccio è che il sistema operativo ospite necessita di modifiche, poiché le chiamate di sistema devono essere sostituite da chiamate all'Hypervisor, "il quale non virtualizza completamente l'hardware, ma ne offre un'interfaccia simile a quella reale. Ne consegue che le performance sono migliori, non essendo più necessaria l'emulazione delle chiamate di sistema. Tuttavia la separazione tra hardware e sistema operativo non è netto come nel caso della full virtualization, e c'è bisogno di cambiare il codice del sistema operativo"[39]. Un esempio di questo tipo di virtualizzazione si trova in Xen (Software di virtualizzazione)

o nei system p di IBM; il sistema che è maggiormente in grado di avvalersene è AIX.

2.2.4 Virtualizzazione Assistita dall'hardware

Questa categoria è più difficile da definire perchè, mentre alcuni lo ritengono un vero e proprio approccio diverso dalla virtualizzazione, che si distingue quindi dalla full virtualization e dalla paravirtualization, altri lo intendono più come un supporto tecnologico alle due modalità di virtualizzazione di cui sopra. "In entrambi i casi esso implica comunque l'adozione di specifiche soluzioni hardware per aumentare le performance dei sistemi virtualizzati. Ad esempio Intel Virtualization Technology (Intel®VT-x[1]) implementa il supporto hardware alla virtualizzazione, accelerando il passaggio di controllo tra il sistema operativo ospitante e quello ospitato, permettendo di assegnare alcuni dispositivi di input o output unicamente al sistema ospitato ed ottimizzando l'uso delle reti"[39]. Altri esempi di questo tipo di tecnologia sono AMD-V, Intel®EPT, VIA VT. Questa soluzione permette di realizzare la virtualizzazione supportandola direttamente dall'hardware.

2.3 Rete

Una tipica architettura fisica di un sistema Cloud si compone di un vasto numero di calcolatori, componenti di memorizzazione di massa e dispositivi di rete, collegati tra loro in maniera strutturata, formando solitamente dei cluster supportati da architetture di rete virtualizzate in grado di adattarsi al contesto applicativo specifico. In questo nuovo scenario, l'infrastruttura di rete assume un ruolo fondamentale al fine di garantire una buona e corretta interconnessione tra i vari elementi allocati. Alla base di un infrastruttura distribuita, tipica del cloud computer, c'è la delocalizzazione dell'informazione; questo fa si che la rete rappresenti la parte abilitante dei servizi Cloud. L'architettura di rete in questo contesto si è evoluta in modo tale da far largo uso delle tecnologie di virtuallizzazione, e dei nuovi protocolli e servizi per il trasferimento delle informazioni. Alla base di una buona soluzione in tal senso vi sono i seguenti punti :

- Virtualizzazione dell'infrastruttura di rete, in cui gli apparati per l'instradamento del traffico sono virtuali e realizzano un'infrastruttura altamente modulare e flessibile

[1] Si possono tovare informazioni in merito alla tecnologia attraverso il link nell'appendice A.1 riga 43

- Integrazione di cloud pubblici e privati, tramite architetture che massimizzano il throughput[2] tra gli elementi singoli, all'interno di un'isola cloud e/o tra diverse isole
- Progetto di reti di interconnessione a banda larga per data center, con tecnologie ottiche ad alta capacità.
- Gestione automatizzata delle risorse di un data center tramite soluzioni integrate multipiattaforma.

2.4 Scalabilità

Per poter garantire la scalabilità del sistema, l'architettura deve essere costruita in modo da poter utilizzare più calcolatori, in grado di condividere il carico di lavoro e fornire un grande livello di disponibilità. Tali architetture, organizzate anch'esse su più livelli, "sono tipicamente costituite da vari elementi, tra i quali si trovano il Load Balancer, che ha il compito specifico di ripartire equamente il carico di lavoro tra i server. I dati costituiscono il punto critico di tali architetture, poiché spesso rappresentano il collo di bottiglia delle applicazioni. Per risolvere questo problema, è possibile attuare politiche di partizionamento, caching e replicazione, in modo da distribuire i dati su più server, ripartendo in tal modo anche il carico di lavoro. Queste scelte non sono a costo zero, in quanto per gestire la consistenza dei dati e mantenere la sincronizzazione tra le varie copie, si genera traffico, e ciò richiede risorse di calcolo (CPU). In quest'ottica è importante definire ed individuare il miglior modello di consistenza applicabile, il quale a sua volta potrebbe influire sul modello dei dati. Un'ultima osservazione che è opportuno fare parlando di architetture di scalabilità geograficamente molto distribuite, è che per poter ridurre la latenza dovuta al tempo di accesso ai dati, sarebbe opportuno mantenere tali dati vicini agli utenti che li utilizzano, soprattutto se si presentano molto statici e cambiano raramente"[39]

2.5 Cloud OS

Il Cloud Operating System, o Cloud OS è la componente che ha come obbiettivo principale quello di fornire l'infrastruttura come un servizio (IaaS), sia per i tenant[3] del Cloud che per i livelli superiori, come PaaS e SaaS. Per

[2]Nell'ambito delle telecomunicazioni, si intende per throughput di un canale di comunicazione, la sua capacità di trasmissione effettivamente utilizzata.

[3]Clienti o Organizzazioni

raggiungere tale obiettivo si deve saper gestire sia l'architettura fisica sottostante che quella virtualizzata, e si devono poter fornire ai client le risorse virtuali richieste. Entriamo nel merito di ciò che deve fornire un Cloud OS: è richiesta un'interfaccia pubblica, che permetta di accedere ai servizi dell'infrastruttura dall'esterno, nascondendo la virtualizzazione ed i dettagli legati alla sua gestione. L'interfaccia deve essere indipendente dall'hardware fisico sottostante, come per i normali sistemi operativi, ed anche dalla tecnologia di virtualizzazione adottata. Oltre a questo sarà necessaria un'interfaccia più ricca, disponibile solo per gli amministratori del sistema, tramite cui essi possano intervenire in maniera più profonda nella gestione del sistema, ad esempio, controllando e monitorando i componenti virtuali e fisici del sistema.

2.5.1 Componenti

Per fornire un'astrazione dell'infrastruttura sottostante, il Cloud OS fa uso di adattatori e driver, che gli permettono di interagire con le varie tecnologie di virtualizzazione. I componenti che costituiscono il nucleo di un Cloud OS si appoggiano a questi driver per gestire, monitorare e distribuire l'infrastruttura virtualizzata. I "componenti principali sono:

- Virtual Machine Manager (VMM), che ha un ruolo centrale, poiché i Cloud OS tipicamente definiscono le macchine virtuali come l'unità base di esecuzione, analogamente ai thread nel caso di sistemi operativi desktop multi-threaded;
- Network Manager, che gestisce la comunicazione tra i componenti, la creazione di reti private e l'assegnazione di indirizzi pubblici;
- Storage Manager, che fornisce l'accesso a funzionalità di memorizzazione persistente delle informazioni;
- Information Manager, che monitora e raccoglie informazioni riguardanti lo stato delle macchine virtuali, i server fisici ed altri componenti.

Un Cloud OS si può inoltre dotare di driver e adattatori che gli permettano di accedere a provider remoti. Il componente che utilizza questi driver è detto Federation Manager, e si occupa di gestire, monitorare e distribuire risorse virtuali remote, provenienti da altri provider"[39].

2.5.2 Proprietà architetturali del Cloud

Mettendo assieme quanto detto fino a questo punto, e quanto descritto nell'articolo Cloud-oriented platforms: bearing on application architecture and

design patterns[30], esporremo di seguito le caratteristiche importanti di una piattaforma Cloud utilizzate per lo sviluppo di applicazioni orientate al Cloud.

2.5.3 Definizione di piattaforma

Una piattaforma è l'insieme di concetti, componenti e paradigmi architetturali che formano il nucleo di un ambiente di sviluppo e distribuzione di applicazioni. Ogni piattaforma ha le sue proprietà architetturali, le quali influenzano le applicazioni che tramite essa vengono sviluppate.

2.5.4 Piattaforme e sviluppo software

Un progettista dovrebbe sempre tenere in considerazione le proprietà sotto elencate, durante il processo di sviluppo del software, poiché esse sono in grado di arricchire il suo prodotto di qualità e caratteristiche che lo aiutano a soddisfare requisiti sia funzionali che non. Analizzeremo ora più nel dettaglio le caratteristiche dell'architettura Cloud, affrontando prima quelle generali delle piattaforme virtualizzate, poi quelle specifiche delle piattaforme Cloud. Da esse dedurremo alcuni principi di base che permetteranno di realizzare applicazioni in grado di sfruttare le potenzialità delle piattaforme Cloud.

2.5.5 Proprietà delle piattaforme virtualizzate

Le principali proprietà delle piattaforme virtualizzate possono essere così sintetizzate:

- Astrazione delle risorse hardware fisiche e delle entità software
- Astrazione dell'interfaccia hardware
- Isolamento degli ambienti di calcolo e multi-tenancy a livello di macchina virtuale
- Scarso isolamento delle performance
- Incapsulamento di tutto il software eseguito su un hardware fisico
- Capacità di controllo dell'hardware limitate da parte delle macchine virtuali
- Possibilità di salvare lo stato di una macchina in un qualsiasi momento, per poterlo successivamente ripristinare

- Possibilità di migrare una macchina virtuale

Molte di queste proprietà hanno recentemente trovato applicazione in maniera innovativa proprio nello sviluppo di piattaforme orientate al Cloud.

2.5.6 Proprietà delle piattaforme Cloud

La ragione che ci ha indotto ad analizzare prima le piattaforme virtualizzate e dopo quelle orientate al Cloud, è che le prime formano il nucleo di tutto il sistema del Cloud computing. Di conseguenza le proprietà elencate nella sezione precedente sono automaticamente incluse anche nelle piattaforme orientate al Cloud. Oltre ad esse possiamo distinguere ulteriori proprietà, alcune condivise da tutte le piattaforme Cloud, altre che si aggiungono in base al servizio offerto (Iaas, Paas o Saas) oppure al tipo di Cloud adottato (Public, Private, Community o Hybrid).

2.5.7 Proprietà generiche

Le proprietà condivise da tutte le piattaforme Cloud sono:

- Natura elastica delle risorse computazionali.
- Capacità di automatizzare l'assegnazione delle risorse.
- Assegnazione delle risorse sulla base della richiesta.
- Misurazione dei servizi erogati.
- Computazione intesa come un servizio accessibile attraverso la rete.
- Mancanza di standard per servizi importanti come la sicurezza o il controllo e la gestione delle macchine virtuali.
- Mancanza di controllo assoluto sulla custodia dei dati e della computazione.
- Tecniche di misurazione non ottimali, derivanti dallo scarso isolamento delle performance delle macchine virtuali.
- Mancanza di meccanismi per il monitoraggio dettagliato delle risorse.
- Difficoltà nel comprendere la struttura delle licenze software, soprattutto negli scenari più complessi, in cui più licenze di software diversi sono applicate in uno stesso ambiente.

2.5.8 Proprietà specifiche dell'Iaas

Di seguito sono elencate le proprietà caratteristiche di una piattaforma Cloud che fornisce servizi secondo il modello Iaas:

- L'utente del Cloud è responsabile dell'installazione e della gestione di tutto il software delle proprie macchine virtuali.
- Possibilità di monitorare l'uso delle risorse e di reagire a differenti tipi di eventi a livello di piattaforma.
- Le applicazioni in esecuzione nella macchina virtuale devono farsi carico di gestire eventi: ad esempio a fronte dell'allocazione di una nuova istanza della macchina, sono le applicazioni che devono sfruttare al massimo le nuove capacità introdotte da tale nuova istanza, suddividendo il carico di lavoro tra le due macchine.
- Controllo limitato sui componenti di rete, ad esempio sui firewall delle macchine ospiti

2.5.9 Proprietà specifiche del Paas

Per quanto riguarda le piattaforme che forniscono servizi di tipo Paas, le principali proprietà in esse individuate sono:

- Nessun controllo sull'infrastruttura sottostante, ovvero sui server, sulla rete, sui sistemi operativi o sul sistema di memorizzazione di massa.
- Possibilità di controllare le applicazioni distribuite ed eventualmente la configurazione del loro ambiente di esecuzione.
- Possibilità di utilizzare solo i linguaggi di programmazione, i tool, le API ed i componenti supportati dal provider: non è possibile, ad esempio, installare nuovo software, aggiornarlo o introdurre un ambiente che non sia tra quelli forniti dal provider.

2.5.10 Proprietà specifiche del Saas

Le piattaforme che forniscono servizi di tipo Saas, sono caratterizzate dalle seguenti proprietà:

- Nessun controllo sull'infrastruttura sottostante.
- Controllo solo su un insieme limitato di impostazioni di configurazione, specifiche per ogni utente.

2.5.11 Proprietà specifiche del Public Cloud

Anche in base a quanto esposto precedentemente, possiamo così riassumere le caratteristiche del Public Cloud :

- L'infrastruttura Cloud è resa disponibile a pagamento, a qualunque utente.
- La proprietà dell'infrastruttura Cloud è detenuta dall'organizzazione che vende i servizi.
- Il Cloud provider detiene la custodia ed il controllo di tutti i dati, delle applicazioni e della computazione ospitati presso la sua infrastruttura.
- L'ambiente di virtualizzazione spesso è omogeneo.

2.5.12 Proprietà specifiche del Private Cloud

Il Private Cloud invece presenta tipicamente le seguenti proprietà:

- Viene utilizzato solamente da un'organizzazione.
- Controllo, custodia e proprietà totali delle applicazioni, dei dati e della computazione.
- Configurazione personalizzata dell'infrastruttura Cloud.
- L'ambiente di virtualizzazione spesso è omogeneo.

2.5.13 Proprietà specifiche del Community Cloud

Le proprietà più rilevanti del Community Cloud si possono così riassumere:

- Infrastruttura Cloud condivisa tra più organizzazioni.
- Supporto di una determinata comunità di organizzazioni, che hanno precedentemente condiviso obbiettivi ed ambiti lavorativi.
- Le organizzazioni appartenenti alla comunità hanno il controllo, la custodia e la proprietà totali delle applicazioni, dei dati e della computazione.

2.5.14 Proprietà specifiche del Hybrid Cloud

Per quanto riguarda Hybrid Cloud, le caratteristiche principali di tali piattaforme sono:

- Struttura logica che unisce due o più Cloud, eventualmente anche di tipo diverso.
- Ogni Cloud che la costituisce rimane un'entità separata, e mantiene le sue proprietà architetturali.

2.6 Evoluzione Sistemi Operativi Cloud

L'evoluzione dei sistemi operativi sta sempre più integrando funzionalità per agevolare il Cloud; in questo contesto quando parliamo di Cloud Operating System intediamo distribuzioni o sistemi concepiti per integrare il più possibile le funzionalità accessibili attraverso il Cloud Computing. Questi sistemi stanno rapidamente acquistando una grande importanza, non solo nel mondo dei Cloud provider, ma anche nei contesti aziendali di molte grandi e medie imprese, che fino a poco tempo fa non avevano alcun interesse o bisogno di rivolgersi al Cloud Computing. Ciò è dovuto al fatto che molte organizzazioni si stanno rendendo conto che centralizzare le proprie risorse informatiche, garantendone una visione omogenea, una gestione di più alto livello ed un modello di utilizzo elastico, caratteristiche tipiche degli ambienti Cloud, può portare ad una notevole riduzione del lavoro necessario al mantenimento e all'uso di tali risorse, quindi ad una riduzione dei costi, legata al miglioramento delle prestazioni. Lo stimolo derivante da questa presa di coscienza, sta portando molte aziende in tutto il mondo ad intraprendere un percorso evolutivo, che le porterà a trasformare i loro data center, eventualmente già virtualizzati, in infrastrutture di tipo Private Cloud. Questa trasformazione porta con sé notevoli vantaggi:

- Divisione tra applicazioni e servizi, ed i relativi server fisici su cui sono in esecuzione;
- Divisione tra i dati ed i dispositivi di memorizzazione di massa in cui sono contenuti;
- Possibilità di estendere l'infrastruttura locale privata con risorse remote provenienti da altre infrastrutture, che hanno subito lo stesso cambiamento, o con risorse provenienti da Public Cloud.

Dal punto di vista dell'amministrazione dell'infrastruttura, ciò si traduce:

- una semplificazione dei processi di gestione dei server per ottenere la potenza desiderata.
- in una grande facilità nel ridimensionamento dell'infrastruttura fisica
- in un maggiore isolamento dei servizi e degli utenti
- possibilità di bilanciare il carico di lavoro automaticamente tra le macchine per migliorarne l'efficienza e l'utilizzo
- semplificata gestione della replicazione dei server per aumentare la tolleranza ai guasti.

Per operare questa trasformazione, organizzando quindi la propria infrastruttura secondo un'architettura di tipo Cloud, è indispensabile la presenza di un Cloud Operating System. In precedenza si è già accennato ad esso, collocandolo all'interno dell'architettura Cloud, spiegandone gli obbiettivi ed elencandone i principali componenti. In questo capitolo i Cloud OS verranno affrontati in maniera più approfondita, in particolare nei suoi componenti, come descritto in Iaas cloud architecture: From virtualized data centers to federated cloud infrastructures[36], con una breve panoramica dei Cloud OS al momento più diffusi.

2.7 Analogie Desktop OS e modularizzazione

Ci sono molte similitudini tra i classici sistemi operativi e quelli Cloud. I primi hanno come scopo principale la gestione delle risorse di un computer; i Cloud OS devono saper gestire le risorse dell'infrastruttura virtualizzata sottostante. I sistemi operativi desktop devono fornire un ambiente di esecuzione sicuro e multi-threaded[4] per le applicazioni utente, che garantisca il loro isolamento; quelli Cloud devono fornire un ambiente di esecuzione sicuro, multi-tenant[5], che garantisca ai servizi utente l'isolamento reciproco. Infine i sistemi operativi desktop devono permettere la condivisione di risorse tra utenti diversi, astraendo dai dettagli dell'hardware sottostante ed offrendo interfacce, sia agli utenti comuni che agli amministratori, per poter interagire con il calcolatore, ciascuno secondo il proprio ruolo; analogamente i Cloud

[4]Un thread o thread di esecuzione, in informatica, è una suddivisione di un processo in due o più filoni o sottoprocessi, che vengono eseguiti concorrentemente da un sistema di elaborazione monoprocessore (multithreading) o multiprocessore o Multicore

[5]Multi-tenant si riferisce ad una architettura software in cui una singola istanza del suddetto software gira su un server ed è utilizzata da più di un cliente/organizzazione (tenant)

OS devono permettere la condivisione di risorse tra i vari tenant, astraendo dalle infrastrutture sottostanti, sia quelle a livello fisico che quelle virtualizzate, fornendo un insieme di API che permetta l'interazione con il Cloud secondo ruoli diversi. Non deve sorprendere quindi che, come i sistemi operativi desktop sono organizzati in moduli, per poter dominare la complessità del problema e per offrire una maggiore flessibilità e configurabilità, i Cloud OS sono organizzati in componenti, ciascuno dei quali si occupa di risolvere un sotto-problema e si colloca in un determinato ambito di lavoro.

2.8 Virtual Machine Manager

I sistemi operativi desktop definiscono i thread come unità d'esecuzione, fornendo molti meccanismi di comunicazione e sincronizzazione, e le applicazioni multi-thread come le entità di base eseguibili dall'utente. I sistemi operativi Cloud invece definiscono come entità di base i servizi virtualizzati che gli utenti o i tenant possono invocare, e le macchine virtuali organizzate solitamente in architetture a più livelli, come unità d'esecuzione, garantendo strumenti di comunicazione e configurazione. Questa architettura concorre a migliorare la scalabilità, poiché è sufficiente aggiungere macchine virtuali al servizio corrente (scalabilità orizzontale), o ridimensionare una macchina virtuale se supportato (scalabilità verticale). Le applicazioni e i servizi sono tra loro isolati, mentre le macchine virtuali appartenenti ad un'applicazione o ad un servizio non lo sono: ad esse infatti vengono fornite reti virtuali ed altri servizi per permettere e gestire la comunicazione. Il Virtual Machine Manager è responsabile dell'intero ciclo di vita delle macchine virtuali, e deve essere in grado di eseguire varie operazioni in base alle azioni degli utenti o alle loro politiche di scheduling. Tali operazioni comprendono la creazione di istanze, la migrazione, la sospensione e la ripresa o lo spegnimento delle macchine virtuali. Inoltre il Virtual Machine Manager deve essere in grado di garantire il rispetto dei Service Level Agreements (SLA)[6] contratti con gli utenti, che solitamente si esprimono in termini di disponibilità delle macchine virtuali. Esso viene quindi arricchito con dei meccanismi che lo rendono capace di individuare i crash delle macchine virtuali, di riallocarle e riavviarle in caso di errore. Le macchine virtuali vengono solitamente definite tramite un insieme di parametri ed attributi, tra i quali ad esempio, il kernel del

[6] Gli SLA sono strumenti contrattuali attraverso i quali si definiscono le metriche di servizio (es. qualità di servizio) che devono essere rispettate da un fornitore di servizi (provider) nei confronti dei propri clienti/utenti. Di fatto, una volta stipulato il contratto, assumono il significato di obblighi contrattuali.

sistema operativo utilizzato, l'immagine del loro disco rigido, la quantità di memoria volatile e la potenza di CPU assegnate.

2.9 Network Manager

Per poter sviluppare servizi ed applicazioni nel Cloud non c'è bisogno solamente di macchine virtuali, ma occorre anche istanziare reti per interconnettere i vari componenti di un servizio, permettendo loro di comunicare, e rendendere il servizio accessibile anche da utenti esterni, se necessario. Il Network Manager deve quindi saper gestire reti private per le connessioni interne tra i componenti dei servizi, e gestire indirizzi IP pubblici per i componenti di front-end accessibili tramite Internet. Poiché anche le reti fornite dal Network Manager sono virtuali, esso deve disporre di meccanismi e politiche, che gli permettano di assegnare correttamente sia gli indirizzi MAC che quelli IP, e deve anche riuscire a garantire l'isolamento del traffico tra varie reti virtuali, anche se queste condividono la stessa infrastruttura di rete fisica.

2.10 Storage Manager

La funzione principale dello Storage Manager è quella di fornire servizi e sistemi di memorizzazione persistente virtuali, secondo le specifiche tipiche del Cloud. Ciò significa che il sistema di storage deve essere rapidamente scalabile, e deve saper crescere automaticamente in base alla necessità degli utilizzatori, deve avere grande disponibilità ed affidabilità, per evitare perdite di dati in caso di errori o crash, deve garantire ottime performance per supportare grossi carichi di lavoro; ad esempio nel caso di applicazioni orientate ai dati o che ne fanno un uso intensivo, lo Storage Manager deve essere semplice da gestire, permettendo agli utenti di astrarre dalla complessità dell'infrastruttura di storage sottostante, che si può comporre anche di dispositivi diversi e tra loro eterogenei. Una questione molto importante che riguarda in generale la memorizzazione persistente di dati in un sistema altamente scalabile e dinamico, è il modello dei dati. Nella seconda metà del 900 in ambito di ricerca erano stati proposti vari modelli, come quello gerarchico o quello reticolare, ma non aveva tardato ad imporsi su tutti gli altri il modello relazionale, dominando per anni la scena in ogni ambito, didattico, di ricerca e aziendale. Esso deve la sua fortuna alla sua espressività ed alla sua intuitività, che gli hanno permesso di diffondersi ovunque, ed alle alte prestazioni offerte dai sistemi che lo adottano, come i Database relazionali interrogati da linguaggi appartenenti alla famiglia dell'SQL. Tali sistemi in-

fatti hanno subito nel corso del tempo una grande attività di studio volta ad ottimizzarne le prestazioni. I linguaggi derivanti dall'algebra relazionale inoltre sono dotati di una grande espressività e permettono di realizzare ed eseguire operazioni anche molto complesse. Tuttavia tali sistemi hanno delle grosse difficoltà ad implementare la scalabilità, ed è per questo motivo che negli ultimi anni sono stati rivalutati tutti gli altri modelli, che ormai da decenni non venivano più presi in considerazione. Ciò ha avuto chiaramente delle ripercussioni anche nello sviluppo di applicazioni che devono essere eseguite nel Cloud, poiché occorre imparare a modellare e gestire i dati secondo i nuovi modelli adottati.

2.11 Image Manager

La gestione delle immagini delle macchine virtuali è molto importante, sia nei data center virtualizzati, sia nei sistemi più propriamente Cloud. Tali sistemi infatti devono essere in grado di gestire una grande quantità di immagini, appartenenti a vari utenti, con sistemi operativi differenti e diverse configurazioni. La gestione deve essere sicura ed efficiente, inoltre i Cloud OS devono anche possedere degli strumenti per gestire i repository di immagini. Le immagini di macchine virtuali vengono caratterizzate da un insieme di attributi, come il nome, la descrizione del contenuto, il livello di condivisione (privata, pubblica o condivisa a pochi), il proprietario e la sua locazione all'interno del repository. Le funzionalità di base per gestire le immagini, includono la creazione di una nuova immagine in un certo repository, la sua cancellazione, la clonazione di un'immagine esistente, l'aggiunta o il cambiamento di determinati attributi, la condivisione di un'immagine da parte di un utente con altri utenti, o la sua pubblicazione per permetterne l'utilizzo da parte di tutti gli utenti.

2.12 Information Manager

Questo modulo è il responsabile del monitoraggio e della raccolta di informazioni riguardanti le macchine virtuali, i server fisici, ed altri componenti appartenenti sia alla struttura fisica che a quella virtuale. Il suo lavoro è essenziale per assicurare che tutti i componenti stiano funzionando correttamente e stiano fornendo buone prestazioni. Il controllo dei server fisici può essere effettuato installando determinati strumenti sui server stessi, che interagiscono con l'Information Manager. Il controllo delle macchine virtuali invece può essere fatto sulla base delle informazioni fornite dall'Hypervisor,

che tuttavia risultano spesso abbastanza scarse, e soprattutto possono variare da un Hypervisor ad un altro, oppure installando anche in questo caso determinati strumenti sulle macchine virtuali; però c'è bisogno del consenso dell'utente proprietario della macchina in questione, e si tratta comunque di una soluzione abbastanza invasiva. L'Information Manager può fornire vari sensori, ognuno dei quali è responsabile di fornire informazioni riguardo aspetti diversi del sistema, come ad esempio il carico di CPU, l'uso della memoria volatile, il numero di processi in esecuzione, l'uso della memoria persistente, il consumo di energia o di banda. Inoltre può permettere di creare sensori personalizzati in grado di monitorare metriche diverse, più specifiche ed adatte al tipo di servizio o applicazione sviluppata.

2.13 Federation Manager

Il Federation Manager è il componente che abilita l'accesso alle infrastrutture Cloud remote, sia che si tratti di infrastrutture private governate da un Cloud Operating System, sia che si tratti di Public Cloud accessibili tramite le loro interfacce, di cui parleremo nelle prossime sezioni. Tale componente deve fornire i meccanismi che permettano di distribuire le risorse presso Cloud remote, gestirle durante l'esecuzione, ritirarle o terminarle quando necessario, monitorarle secondo i parametri desiderati, autenticare gli utenti che vi accedono e creare immagini di macchine virtuali, le quali probabilmente avranno formati diversi. In base alle capacità offerte dal Cloud remoto, che determinano il livello di interoperabilità raggiungibile nell'interazione tra i due Cloud, può essere possibile fornire meccanismi anche più avanzati, come quelli per la migrazione di macchine virtuali attraverso le diverse infrastrutture, o quelli per la creazione di reti e sistemidi storage condivisi. Il Federation Manager deve essere implementato come un componente interno del Cloud OS; per poter supportare la federation a livello di infrastruttura è tuttavia possibile appoggiarsi a dei servizi esterni, come il progetto Aeolus[7], per raggiungere la federation a livello di utente. Nelle sezioni successive verrà approfondito il concetto di federation, spiegando quali architetture esistono e quali sono i livelli di interoperabilità raggiunti da esse.

[7]Si possono tovare informazioni in merito al progetto attraverso il link nell'appendice A.1 riga 55 e 56

2.14 Scheduler

In un'infrastruttura Cloud esistono due livelli di scheduling: il primo livello stabilisce quali macchine virtuali mandare in esecuzione e quali processori assegnare ad ogni macchina virtuale, e viene gestito dallo scheduler dell'Hypervisor; il secondo livello è responsabile di decidere su quali server fisici distribuire le macchine virtuali. La schedulazione si basa su diversi criteri di allocazione e migrazione delle macchine virtuali. I criteri sono i seguenti:

- Riduzione del numero di server in uso, per minimizzare il dispendio energetico. Le nuove macchine virtuali dovrebbero essere allocate, quando possibile, presso server già attivi.
- Bilanciamento del carico di lavoro, per evitare la saturazione di alcuni server, con conseguente crollo delle performance; in tal caso le macchine virtuali dovrebbero essere equamente distribuite tra tutti i server disponibili.
- Bilanciamento dell'uso della CPU, che ha lo stesso obbiettivo del punto precedente, ma applica la politica secondo la quale le macchine virtuali devono essere allocate presso i server con la più alta percentuale di CPU libera.
- Bilanciamento termico, per evitare il surriscaldamento dei server riducendo i meccanimi di raffredamento.

In questo caso le macchine virtuali dovrebbero essere allocate presso i server con la temperatura più bassa. Come si può facilmente notare spesso i criteri sono in contrasto tra loro, ed un buon scheduler deve riuscire a tenerli tutti in considerazione trovando il giusto equilibrio. Ovviamente lo scheduler non è incaricato solamente di allocare i server fisici alle macchine virtuali nel momento della creazione o della loro accensione, ma deve costantemente monitorare la situazione, per decidere quando è necessario migrare le macchine virtuali già in esecuzione. In presenza di contesti in cui è disponibile la federation, lo scheduler può anche decidere di allocare o migrare le macchine virtuali presso server remoti. Infine lo scheduler deve anche tener conto di ulteriori vincoli imposti dagli utenti, come i livelli minimi di CPU e memoria volatile, allocati alle proprie macchine; la necessità di tenere due o più macchine virtuali in esecuzione sullo stesso server, nei casi in cui serve minimizzare il traffico di rete, se tali macchine interagiscono molto frequentemente, oppure quando ci sono vincoli legati alla distribuzione geografica, che sorgono spesso a causa delle differenti leggi vigenti nei vari paesi, riguardanti il trattamento di dati personali, e vincoli legati al rispetto dei Service-level Agreements, esprimibili

come capacità di CPU garantita ed affidabilità dei servizi. Lo scheduler viene invocato ogni volta che una nuova macchina virtuale è in attesa che le venga allocato un server fisico, e periodicamente per eseguire le ottimizzazioni necessarie sull'intera infrastruttura virtuale, riallocando, se necessario, alcune macchine virtuali. Per poter meglio garantire l'operatività, questo modulo deve poter interagire con il Virtual Machine Manager ed eventualmente, con il Federation Manager.

2.15 Service Manager

Un Cloud OS deve essere in grado di gestire servizi virtualizzati, spesso costruiti su un'architettura a più livelli, e costituiti da più componenti legati tra loro da relazioni di dipendenza. I servizi vengono resi disponibili distribuendoli tipicamente su più macchine virtuali, che a loro volta ereditano o generano nuove relazioni di dipendenza, in base ai componenti o livelli che ospitano. Inoltre i servizi solitamente necessitano di appoggiarsi ad elementi di memorizzazione persistente e di comunicazione. Una delle funzionalità del Service Manager, la prima ad entrare in gioco a fronte della realizzazione di un nuovo servizio, è il controllo che viene effettuato sul servizio stesso, per determinare se può essere accettato nel Cloud o se deve essere rifiutato, in base alle risorse che richiede, alle autorizzazioni di cui necessita, ed alle disponibilità del Cloud. Una volta che un servizio è stato accettato, il Service Manager diventa responsabile del suo intero ciclo di vita, che include operazioni come il deployment, la sua sospensione o riattivazione, e la sua cancellazione. Per poter effettuare il deployment del servizio, il Service Manager deve interagire con lo scheduler, per poter determinare correttamente dove distribuire le macchine virtuali di cui tale servizio necessita. Un'altra importante funzionalità del Service Manager è la gestione dell'elasticità dei servizi. Il Service Manager per poter adempiere a questo compito, incorpora solitamente vari meccanismi per innescare la scalabilità automatica dei servizi, i quali si basano su regole precise che determinano come scalare il servizio (verticalmente o orizzontalmente) e quando (in base al superamento di alcune soglie su determinate metriche imposte dall'utente, o in maniera automatica).

2.16 Interfacce

Le funzionalità del Cloud OS possono essere rese pubbliche ed esposte agli utenti, tramite determinate interfacce. Queste interfacce costituiscono ciò

che l'utente del Cloud percepisce effettivamente come Infrastructure as a Service, poiché determinano l'insieme dei servizi di cui l'utente può disporre per interagire con l'infrastruttura. Ogni Cloud provider fornisce le proprie API con strumenti diversi. La modalità più diretta, ma anche più complessa, per richiedere i servizi del Cloud, è tramite le richieste del protocollo HTTP, che è diventato ormai lo standard per l'interazione secondo l'architettura SOA[8]. Tuttavia, per semplificare lo sviluppo di applicazioni e per fare in modo che le richieste vengano rappresentate in una modalità più ricca di significato per i linguaggi di programmazione, spesso vengono anche fornite della librerie specifiche per ogni linguaggio supportato. Per permettere l'utilizzo dei servizi direttamente agli utenti, piuttosto che delle applicazioni, vengono anche forniti dei tool, sia grafici sia da linea di comando. Il problema tuttavia è che, nonostante alcune di queste interfacce siano talmente diffuse e ben progettate che stanno ormai diventando degli standard de facto, come le API di Amazon Web Services, che vengono infatti supportate nativamente dal Cloud OS Eucalyptus, il livello di eterogeneità è ancora molto alto, e ciò rende molto complessa l'interoperabilità e la portabilità attraverso Cloud diverse. Esistono oggi due approcci diversi che cercano di risolvere il problema dell'eterogeneità: i cosiddetti Cloud adapters, come Apache Deltacloud o Libcloud, che prendono in considerazione un vasto numero di Cloud tra i più diffusi e cercano di realizzare uno strato intermedio, che permetta la comunicazione; o gli organismi di standardizzazione, che cercano invece di stabilire delle interfacce comuni, in grado però di permettere ai Cloud provider di offrire i loro servizi, eventualmente anche differenziandosi gli uni dagli altri. Tra gli standard che attualmente si stanno diffondendo e stanno acquistando sempre più visibilità e importanza, ci sono OGF OCCI,descritto anche in [33], DMTF CIMI e OVF.

2.17 Autenticazione e autorizzazione

Come per ogni ambiente condiviso, anche per il Cloud è importante possedere dei meccanismi in grado di riconoscere ed autenticare gli utenti e gli amministratori, per poter anche fornire loro accesso alle risorse, per le quali dispongono delle autorizzazioni necessarie. I componenti che si occupano dell'autenticazione hanno come obbiettivo proprio quello di verificare e con-

[8]Nell'ambito dell'informatica, con la locuzione inglese di Service-Oriented Architecture (SOA) si indica generalmente un'architettura software adatta a supportare l'uso di servizi Web, per garantire l'interoperabilità tra diversi sistemi, così da consentire l'utilizzo delle singole applicazioni come componenti del processo di business e soddisfare le richieste degli utenti in modo integrato e trasparente

fermare l'identità dell'utente, che cerca di accedere alle risorse del Cloud. I metodi che possono essere utilizzati per implementare queste funzionalità sono svariati, e vanno dalla semplice verifica di una password, a protocolli molto più complessi, che oggi si stanno largamente diffondendo a causa dell'importanza crescente della sicurezza nell'ambito del Cloud, e in generale nei sistemi altamente distribuiti ed aperti. Le politiche di autorizzazione invece, sono responsabili del controllo e della gestione dei privilegi e dei permessi dei vari utenti riguardanti l'accesso di ogni tipo di risorsa, dalle macchine virtuali, alle reti ed ai sistemi di storage. Il controllo degli accessi può essere implementato tramite meccanismi basati sul concetto di ruolo. Il ruolo viene definito come un insieme di permessi per svolgere determinate operazioni, su alcune specifiche risorse. Ad ogni utente possono quindi essere assegnati più ruoli. In aggiunta si possono usare dei meccanismi, che limitano l'ammontare di risorse che un utente può utilizzare, come la larghezza di banda, la quantità di memoria, e la potenza di calcolo disponibile.

2.18 Accounting ed auditing

Lo scopo dei componenti che si occupano dell'accounting è di ottenere informazioni riguardanti l'uso di determinate risorse da parte dei servizi. Il componente che svolge l'attività di Accounting, si affida al Federation Manager per accedere alle informazioni riguardanti le metriche da quest'ultimo monitorate; inoltre è essenziale poter implementare meccanismi di allocazione dei costi, in maniera tale che ad ogni utente sia addebitato il pagamento delle sole risorse utilizzate dalle sue applicazioni. Un altro aspetto fondamentale degli ambienti Cloud è l'auditing: esso consiste nella raccolta delle informazioni riguardanti l'attività delle varie risorse, in particolare chi ne ha ottenuto l'accesso, quando l'ha ottenuto, e che tipo di utilizzo ne ha fatto. Queste informazioni risultano utili per migliorare la sicurezza del Cloud e per proteggerlo da varie minacce, come accessi non autorizzati, utilizzo abusivo delle risorse, ed intrusioni illecite.

2.19 Livello di accoppiamento di Cloud

Come già accennato nella sezione relativa al Federation Manager, approfondiamo ora il tema della federazione, analizzando in particolare il grado di interoperabilità che è raggiungibile da due Cloud diversi. Esso viene determinato in funzione delle possibilità di interagire che i Cloud si offrono reciprocamente, e comprende aspetti come il controllo ed il monitoraggio di

risorse remote, l'istanziazione di reti che attraversino i confini dei Cloud, e la migrazione di macchine virtuali. In base agli scenari più diffusi, è possibile classificare i livelli di accoppiamento in tre principali categorie:

- Interoperabilità debole. "Lo scenario tipico in questo caso è costituito da istanze di Cloud indipendenti, come per Cloud privati che si espandono appoggiandosi ad un Cloud pubblico, caratterizzati da una scarsa interazione. Solitamente non c'è, o è molto debole, il controllo sulle risorse remote; il monitoraggio delle risorse è assai limitato, e le funzionalità più avanzate, come la migrazione di macchine virtuali, non sono supportate. In questo caso, ad esempio, un Cloud può non avere alcun controllo sulla localizzazione delle macchine virtuali dell'altro, e può ricevere informazioni solo riguardanti le metriche più importanti e basilari, come l'uso della CPU e della memoria da parte di una macchina virtuale"[39].

- Interoperabilità media. "In questo scenario solitamente sono presenti Cloud le cui organizzazioni hanno stipulato qualche forma di contratto, stabilendo a priori sotto quali termini e condizioni un Cloud può utilizzare le risorse dell'altro. Tale contratto può abilitare un controllo più sofisticato sulle risorse remote, ed un loro più fine monitoraggio; ad esempio, possono essere fornite informazioni sulla localizzazione ed il consumo energetico delle risorse, e può esserci la possibilità di intervenire in qualche modo su questi aspetti. Inoltre è possibile creare delle reti virtuali che attraversano i confini dei Cloud"[39].

- Interoperabilità forte. "In questo caso solitamente i Cloud appartengono tutti alla stessa organizzazione, e vengono gestiti dallo stesso Cloud OS. Possono quindi essere garantiti controlli molto avanzati sulle risorse remote, ed un monitoraggio molto dettagliato su tutte le metriche disponibili. Sono talvolta supportate anche le funzionalità più avanzate, come la migrazione di macchine virtuali, o la creazione di sistemi di storage virtuale, che attraversino i confini dei Cloud. Il livello di accoppiamento può essere elevato al punto da poter definire tale federazione, almeno da un punto di vista esterno, come un unico Cloud, non potendo percepire le differenze ed i confini tra i Cloud che la compongono"[39].

2.20 Architetture di federazione

I livelli di accoppiamento analizzati nella sezione precedente, trovano riscontro "nelle architetture tipicamente istanziate con lo scopo di ottenere la feder-

azione tra Cloud. Tali architetture non sono ancora degli standard uniformemente riconosciuti, ma sono state dedotte studiando gli scenari di federazione più diffusi oggi nel mondo del Cloud. Possono essere sintetizzate in quattro categorie principali"[39].

- **Cloud bursting.** "Quest'architettura è già stata analizzata quando abbiamo affrontato i principali modelli di Cloud: si tratta infatti di un Cloud ibrido. Essa viene istanziata quando c'è la necessità di integrare ed espandere un'infrastruttura esistente, non necessariamente organizzata a sua volta come un Cloud, appoggiandosi ad un Cloud pubblico. Il livello di accoppiamento raggiunto in questo caso è debole, poiché il Cloud OS remoto interagisce con quello locale come se fosse un qualunque client"[39].

- **Cloud broker.** "In questo caso l'architettura fa uso di un componente particolare che funge da mediatore, o broker appunto, il quale ha accesso a diversi Cloud pubblici. Nella sua implementazione più semplice, deve essere in grado di gestire le risorse degli utenti nei Cloud che essi desiderano. Broker più complessi sono anche in grado di applicare determinate politiche di scheduling, basate su criteri di ottimizzazione, ad esempio dei costi o delle prestazioni, decidendo su quali Cloud distribuire i servizi, o eventualmente i singoli componenti. Poiché tipicamente i Cloud provider non consentono la gestione avanzata delle proprie risorse da parte di utenti esterni, anche in questo caso il livello di interoperabilità raggiungibile dai vari Cloud coinvolti, è piuttosto basso. Esempi di Cloud broker attualmente disponibilisono BonFIRE, Open Cirrus o FutureGrid"[39].

- **Aggregazione di Cloud.** "L'aggregazione di Cloud viene realizzata a partire da due o più Cloud che interagiscono unendo alcune delle proprie risorse, in modo da fornire agli utenti un'infrastruttura virtuale più grande. L'interoperabilità in questo caso è medio livello, poiché le organizzazioni possono fornirsi vicendevolmente qualche forma di controllo avanzato, essendo in grado di distinguere le interazioni provenienti dal Cloud aggregato da quelle degli utenti comuni. Solitamente per poter realizzare tale architettura, è necessario stipulare un contratto e stabilire i termini secondo i quali è possibile interagire, come già visto nella sezione precedente, analizzando il grado di accoppiamento che corrisponde a questa architettura. Sono tuttavia possibili delle aggregazioni di Cloud in grado di cooperare più profondamente: ciò avviene quando i Cloud coinvolti appartengono tutti alla stessa or-

ganizzazione. Un esempio di architettura di aggregazione di Cloud è RESERVOIR"[39].

- **Architettura multi-livello.** "Questo tipo di architettura consiste in due o più Cloud, ciascuno dei quali possiede il proprio Cloud OS, che vengono a loro volta gestiti da un altro Cloud OS, posto gerarchicamente ad un livello superiore. Poiché solitamente questa architettura viene istanziata da Cloud provider che possiedono più infrastrutture separate, disposte geograficamente in regioni spesso molto distanti, per garantire tolleranza ai guasti e riduzione delle latenze, il livello di interoperabilità raggiungibile è il massimo. Il Cloud OS a livello superiore, ha un controllo completo delle risorse dei vari Cloud, e ne fornisce un'interfaccia agli utenti tale da far loro percepire la presenza di un unico Cloud sottostante"[39].

Concludiamo questa sezione osservando che, mentre per le architetture che forniscono un livello di accoppiamento debole è stato finora fatto molto lavoro e sono ora disponibili anche sul mercato molte utili soluzioni, per quanto riguarda le altre architetture è forse necessario progredire ulteriormente con lo sviluppo, soprattutto in merito a quelle multi-livello, che sono finora utilizzate solamente secondo la modalità sopra esposta.

2.21 Breve analisi di Cloud OS esistenti

In questa sezione verrà offerta una panoramica generale dei Cloud OS più diffusi OpenStack, OpenNebula ed Eucalyptus.

2.21.1 OpenStack

"OpenStack è un sistema operativo open source, che permette di realizzare sia Private che Public Cloud. Questo progetto è stato cominciato nel 2010 da Rackspace Cloud e NASA, che vi hanno fatto confluire alcuni altri loro progetti, tra cui Swift, Nebula e Nova, che sono successivamente diventati parte di OpenStack, talvolta addirittura trasformandosi in alcuni suoi moduli. Tale Cloud OS infatti è dotato di un'architettura modulare, come mostrato in (fig.2.2), i cui componenti principali verranno ora analizzati"[39].

Dashboard

"Il modulo chiamato **Horizon** fornisce agli utenti ed agli amministratori un'interfaccia grafica per accedere, fornire ed automatizzare le risorse. Es-

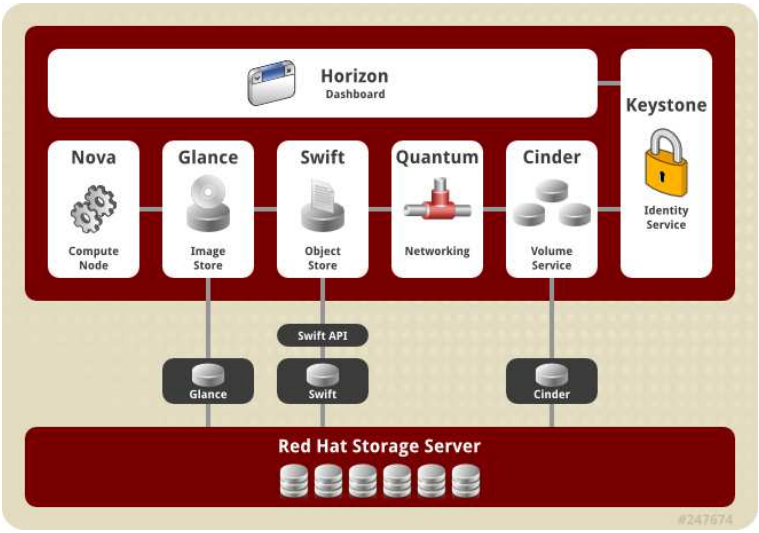

Figura 2.2: Openstack Moduli

sa è progettata in modo da essere estensibile, permettendo l'installazione di software fornito da altri in modo da aumentarne le funzionalità, ad esempio tramite tool che permettano di gestire il monitoraggio o l'allocazione dei costi. Essa è implementata come un'applicazione Web ed è rivolta sia agli utenti, per la gestione delle proprie risorse, sia agli amministratori, ai quali fornisce una visione d'insieme dell'intero Cloud. Tramite essa inoltre gli utenti possono rifornirsi automaticamente di nuove risorse, rispettando i limiti che gli amministratori possono imporre, sempre grazie all'uso della Dashboard"[39].

Compute

OpenStack permette ai provider di offrire risorse computazionali on-demand gestendo grandi reti di macchine virtuali. Tali risorse sono accessibili tramite delle API da parte della applicazioni, e tramite un'interfaccia Web da parte degli utenti ed amministratori. Questo modulo, chiamato **Nova**, supporta gran parte delle tecnologie di virtualizzazione attualmente esistenti; gli hypervisor consigliati in particolare sono XenServer vSphere e KVM. L'architettura di computazione è progettata per scalare orizzontalmente con grande facilità, appoggiandosi ad hardware di qualità ordinaria, senza la necessità di utilizzare quindi macchine molto costose.

Image Store

OpenStack Image Service è il progetto che gestisce i servizi di scoperta, registrazione e recupero delle immagini delle macchine virtuali; tramite delle API di tipo ReSTful[9], **Glance** permette sia di interrogare i metadati delle immagini, che il recupero delle immagini stesse. Glance fornisce numerosi metodi per l'immagazzinamento delle immagini, dall'uso di un path sul filesystem locale, all'utilizzo di un servizio di storage come **Swift**. Le immagini immagazzinate possono essere in molti formati, tra cui RAW, AMI, QCOW2, VMDK, OVF, in modo da essere compatibili con gli standard dei vari hypervisor. Oltre alla possibilità di fare l'upload delle immagini delle macchine virtuali, glance offre anche la possibilità di gestire le copie di backup dei dischi delle VM. Per ogni immagine presente nell'archivio, glance assegna un codice univoco, chiamato uuid, e ne associa uno stato a seconda dell'azione che sta eseguendo (queued, saving, active, killed, deleted, pending_delete).

Networking

Questo componente, chiamato **Quantum**, è un sistema altamente scalabile e controllabile tramite API per la gestione di reti ed indirizzi IP. Esso è progettato prestando particolare attenzione agli aspetti che permettono di fare in modo che le reti non diventino il collo di bottiglia o, in generale, una limitazione per l'infrastruttura virtuale o le applicazioni. Quantum permette sia l'assegnazione di indirizzi IP statici che l'adozione di protocolli di configurazione automatica, come il DHCP. Inoltre i cosiddetti indirizzi flottanti, che possono essere agilmente riassegnati, permettono di redirigere il traffico da una macchina ad un'altra. Inoltre esso è dotato di funzionalità aggiuntive che gli permettono di fornire ulteriori servizi, come il bilanciamento del carico di lavoro, l'individuazione di intrusioni, l'istanziazione di Virtual Private Network (VPN) o la creazione di firewall.

Storage

I servizi di storage offerti da OpenStack, si suddividono in due moduli. Il primo, **Swift**, fornisce una piattaforma di Storage a oggetti distribuita ed accessibile tramite API, che può quindi essere perfettamente integrata nelle applicazioni, oppure usata per questioni come il backup o l'archiviazione di

[9]REST definisce un insieme di principi architetturali per la progettazione di un sistema. Rappresenta uno stile architetturale, cioè non si riferisce ad un sistema concreto e ben definito, nè si tratta di uno standard stabilito da un organismo di standardizzazione.

informazioni. Il secondo invece, chiamato **Cinder**, permette di gestire dispositivi di memorizzazione di massa virtualizzati, connettendoli a specifiche macchine virtuali per aumentarne le capacità e le prestazioni. Inoltre, quest'ultimo modulo permette anche di gestire facilmente gli snapshot dei dispositivi, facendo in modo che essi possano essere ripristinati oppure replicati.

Identity Service

OpenStack Identity Service si occupa di gestire un elenco centralizzato degli utenti e dei privilegi che essi hanno sull'intera infrastruttura. Oltre a mantenere tale elenco, il **Keystone** mantiene un catalogo di tutti i servizi implementati sull'infrastruttura di cloud computing, assumendo così un ruolo fondamentale per tutti i componenti che devono interagire. Il progetto è stato sviluppato in modo che possa supportare differenti sistemi di autenticazione, tra i quali il classico utilizzo di una coppia nome utente e password. La sua architettura modulare permette di estendere i possibili metodi di autenticazione, aggiungendo moduli scritti secondo le proprie esigenze, come nel caso del modulo per l'autenticazione basata sui certificati X.509 nelle cloud federate. L'organizzazione e l'assegnazione dei privilegi viene gestita utilizzando tre strutture principali denominate users, tenants e roles. La struttura degli users mantiene l'elenco delle utenze conosciute dal sistema, trattando in maniera analoga utenze dedicate ad utenti reali, e utenze dedicate all'accesso dei servizi; la struttura dei tenants serve a raggruppare gli utenti e i servizi in un contesto comune, permettendo così di associare ruoli differenti agli stessi utenti in diversi tenants. Questa architettura permette di avere una gestione molto flessibile e allo stesso tempo specifica dei privilegi, creando dei sistemi denominati multi-tenants, dove gli utenti possono acquisire privilegi differenti su diversi sistemi.

2.21.2 Eucalyptus

Eucalyptus (fig.2.3) è un software open source che permette di realizzare Hybrid Private Cloud. Esso fornisce dei servizi di tipo IaaS che vengono esposti come servizi Web. Le sue API sono sviluppate in modo da essere compatibili con quelle di Amazon Web Services, chiamato anche AWS, un Cloud provider che fornisce anch'esso servizi di tipo IaaS. Grazie a questa scelta progettuale è possibile raggiungere un ottimo grado di portabilità tra le due diverse piattaforme. Ad esempio, un'applicazione scritta per AWS dovrebbe essere in grado di funzionare, se necessario con qualche piccolo cambiamento, anche su una piattaforma governata da Eucalyptus. Inoltre anche molti dei tool sviluppati per AWS, sono compatibili con questo Cloud OS. Euca-

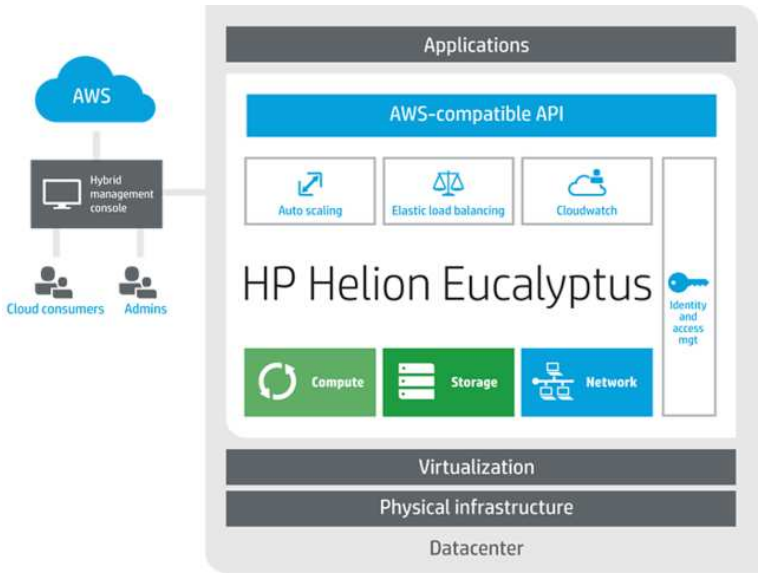

Figura 2.3: Architettura Eucalyptus

lyptus può essere configurato in modo da offrire servizi ad alta affidabilità (high-availability o HA), per adottare automaticamente dei meccanismi per il recupero dagli errori. Quando questa modalità viene attivata, ogni componente o servizio di Eucalyptus viene copiato in una replica, detta passiva. In caso di errore o fallimento del servizio attivo, quello passivo prende il controllo delle operazioni che l'altro stava svolgendo, sostituendolo nel suo lavoro di gestione di alcuni aspetti del Cloud OS. Molti dei sistemi operativi Windows e Linux sono supportati dalle macchine virtuali di Eucalyptus, le quali possono essere memorizzate tramite vari formati:

- Eucalyptus Machine Image (EMI), che è il formato originale di Eucalyptus.
- Amazon Machine Image (AMI), che è il formato adottato da AWS per le proprie macchine virtuali.
- Formati caratteristici di VMware che possono essere gestiti, convertendoli in uno dei due formati precedenti.

Dal punto di vista della virtualizzazione, Eucalyptus non si occupa direttamente di questo aspetto, ma supporta molti degli ambienti ed hypervisor oggi più diffusi per virtualizzare l'infrastruttura fisica sottostante. L'interfacciamento al sistema da parte degli amministratori può svolgersi completamente per mezzo di uno strumento dotato di interfaccia grafica, Eucalyptus Dashboard, che permette di svolgere le attività tipiche di gestione e supervisione di

un'infrastruttura Cloud, tra cui la configurazione ed il monitoraggio. Si possono inoltre specificare delle risorse sia virtuali che fisiche, oppure delle quote massime, utilizzabili da vari utenti o gruppi di utenti, riguardanti determinate risorse e metriche, come la memoria allocabile o il traffico di rete. Anche i meccanismi di pagamento tipici del modello pay-per-use, sono originariamente supportati. Eucalyptus infine supporta dei sistemi di autenticazione ed autorizzazione degli utenti.

2.21.3 OpenNebula

OpenNebula è un Cloud OS completamente open source, rilasciato con licenza Apache. Esso presenta buone caratteristiche di interoperabilità, poiché permette di scegliere quali interfacce adottare da un insieme piuttosto vasto, che comprende sia standard ufficiali, come OGF OCCI, sia standard de-facto, come le API di AWS. Permette di interagire con il sistema sia tramite un'interfaccia a linea di comando (Command Line Interface, o CLI) che tramite un tool che mette a disposizione un'interfaccia grafica, chiamata Sunstone. Oltre a questo è disponibile in OpenNebula anche un Marketplace per immagini di macchine virtuali già pronte, chiamate Virtual Appliance, che possono essere subito avviate senza bisogno di ulteriori configurazioni per renderle operative nell'ambiente di tale Cloud OS. OpenNebula mette anche a disposizione dei servizi di integrazione che permettono di facilitare la migrazione di componenti da un Cloud all'altro, funzionalità utile soprattutto in presenza di Hybrid Cloud realizzati per ottenere un'estensione delle capacità di un Private Cloud. L'autenticazione degli utenti può avvenire tramite password, oppure altri meccanismi più complessi. Le tecnologie di virtualizzazione supportate comprendono Xen, VMware e KVM. Il monitoraggio può essere effettuato sia tramite gli strumenti forniti da OpenNebula, sia tramite dei tool esterni che quest'ultimo supporta, come Ganglia[10]. Viene infine offerto supporto, per quanto riguarda l'aspetto dei Database, alle tecnologie MySQL e SQLite.

2.22 Conclusioni

In questo capitolo è stato esposto quello che al momento è a disposizione dei provider per implementare un Cloud; tali strumenti sono comunque in continua evoluzione, quindi vi è un'intensa attività di sviluppo/customizzazione, per poter diversificare e ampliare le possibilità date da tali ambienti. In questo contesto Microsoft ha proposto per azureus un framework, che fa

[10]Ganglia è un sistema di monitoraggio scalabile e distribuito per systemi high-performance come cluster e grid. Link del sito ufficiale A.1 riga 54

uso di un paradigma ad attori, che meglio domina la complessità degli ambienti altamente distribuiti come quelli Cloud, e sgrava il programmatore dalle tipiche problematiche connesse a tali ambienti, consentendogli di porre maggiore attenzione all'applicazione sviluppata. Nel seguente capitolo verrà presentato in dettaglio questo nuovo framework proposto da microsoft.

Capitolo 3

Nuovi modelli e paradigmi Orleans

Introduzione

Il paradigma Object-Oriented non è pienamente in grado di dominare la complessità degli ambienti altamente distribuiti come quelli Cloud. Per poterlo utilizzare è quindi necessario dotarsi di librerie ed estensioni dei linguaggi, che permettano di risolvere molti dei problemi che tipicamente sorgono in tali ambienti, legati ad aspetti come ad esempio il multi-threading, la concorrenza, la consistenza degli stati, la sincronizzazione e la coordinazione. Così facendo però spesso si perviene ad un software molto complesso, che finisce col porre un accento maggiore sui problemi di cui sopra, piuttosto che sul contesto applicativo. Una soluzione a queste problematiche può essere ottenuta adottando dei diversi paradigmi, che si adattano molto meglio agli ambienti distribuiti e concorrenti. Uno di questi paradigmi è quello ad attori, che verrà introdotto nella prima sezione di questo capitolo, basandosi sulla descrizione fornita in Actors[34]. Successivamente prenderemo in esame Orleans, un framework tuttora in fase di sviluppo da parte di Microsoft che, adottando un modello molto simile a quello ad attori, permette di scrivere applicazioni affidabili, scalabili ed elastiche. Esso si basa sulla tecnologia .NET ed è reso disponibile ai programmatori, tramite delle librerie per i linguaggi supportati da tale tecnologia, come C# o F#. Le informazioni riguardanti Orleans presenti in questo capitolo sono state tratte da Orleans: cloud computing for everyone[31].

3.1 Il modello ad attori

3.1.1 Introduzione

Il modello ad attori è un modello di programmazione concorrente per lo sviluppo di sistemi paralleli, distribuiti e mobili. L'entità fondamentale di questo modello è l'attore, un oggetto autonomo in grado di operare concorrentemente ed in maniera asincrona rispetto agli altri attori, con i quali comunica solo tramite scambio di messaggi, senza condividerne lo stato. Un sistema realizzato tramite questo modello è una collezione di attori, alcuni dei quali scambiano messaggi con attori o altre entità esterne al sistema. Nel modello Object-Oriented gli oggetti incapsulano informazioni e comportamento, separando così la propria interfaccia dalla implementazione, permettendo una modularizzazione delle applicazioni. Il modello ad attori estende queste caratteristiche alla programmazione concorrente, incapsulando anche il flusso di controllo.

3.1.2 Gli attori

Ogni attore ha un nome in grado di identificarlo univocamente all'interno del sistema, ed un comportamento che ne determina le azioni svolte. Essi possono inviare e ricevere messaggi in maniera asincrona. Per poter inviare un messaggio ad un altro attore, il mittente deve conoscerne il nome, mentre per quanto riguarda la ricezione dei messaggi, essa avviene tramite una mailbox, ovvero una coda di messaggi. Quando un attore prende in carico un messaggio, esegue le operazioni ad esso associate e definite dal suo comportamento, espresso tramite dei metodi analoghi a quelli del paradigma ad oggetti. Il ciclo di vita di un attore prevede che quando ha terminato di eseguire le operazioni associate ad un messaggio, e si trova in stato di idle, se è presente un nuovo messaggio nella propria mailbox, lo legga ed esegua la relativa computazione. Se invece, entra in stato di idle e non è presente alcun messaggio nella mailbox, l'attore si blocca cambiando lo stato in waiting, fino all'arrivo di un nuovo messaggio. In questo modello è importante che l'arrivo di un nuovo messaggio non interrompa mai l'elaborazione che un attore sta svolgendo: essa è sempre portata a termine prima di accettare un nuovo messaggio. Gli attori non possono comunicare in altro modo se non con lo scambio di messaggi; in particolare, ciò significa che non c'è condivisione di stato tra gli attori, quindi le computazioni parallele non presentano problemi di concorrenza, non potendo interferire tra loro in alcun modo.

Le classi di operazioni che un attore può compiere durante la computazione sono solo tre:

- invio di un messaggio;
- modifica del proprio stato;
- creazione di un nuovo attore.

Il modello ad attori introduce il non determinismo: i messaggi infatti vengono scambiati lungo la rete, la quale ha dei ritardi che non sono noti a priori, e possono seguire percorsi diversi; inoltre il modello adotta il parallelismo e la concorrenza. Questi fattori portano alla non conoscenza dell'ordine di arrivo dei messaggi. Un errore comune, quando si programma con il modello ad attori infatti consiste nel supporre che messaggi inviati sequenzialmente da uno stesso attore con un determinato ordine, mantengano tale ordine anche nella ricezione. Questo non è mai vero, nemmeno se tutti i messaggi vengono inviati allo stesso destinatario: nulla infatti impedisce ad un qualunque messaggio di arrivare prima di un altro che era stato spedito precedentemente; ricordiamo infatti che l'invio di messaggi è eseguito in modalità asincrona, ovvero non bloccante, e non si attende nessun tipo di conferma, nè dall'ambiente di esecuzione, nè dal destinatario.

3.1.3 Programmi ad attori

All'inizio dell'esecuzione di un programma ad attori, tutte le mailbox degli attori presenti sono vuote, e gli attori sono tutti in attesa. Perché l'esecuzione abbia effettivamente inizio, è necessario che almeno un attore riceva un messaggio da parte dell'ambiente esterno: questo rappresenta l'entry point del programma. La terminazione di un programma ad attori avviene invece quando tutti gli attori creati sono in stato di attesa, e l'ambiente non è più abilitato ad inviare nuovi messaggi al sistema. Tuttavia è anche possibile che un programma ad attori non termini mai, come può succedere per molte applicazioni interattive, ad esempio sistemi operativi o server di vario genere.

3.1.4 Sincronizzazione

A causa della semantica non bloccante dell'operazione di invio dei messaggi, è necessario introdurre dei meccanismi che permettano di gestire la sincronizzazione. Ciò è fatto sempre tramite la comunicazione.

Remote Procedure Call

Adottando uno stile di comunicazione simile a quello del Remote Procedure Call, si introduce un'operazione di invio di messaggi bloccante. Nello scenario

tipico dell'RPC infatti il chiamante invia una richiesta, e rimane in attesa di una risposta dal destinatario, senza eseguire altre operazioni. Non è corretto rendere tale semantica l'unica disponibile poiché sarebbe in forte contrasto con il parallelismo degli ambienti distribuiti, e impedirebbe a quest'ultimo di essere sfruttato; talvolta si rende però necessario come meccanismo di sincronizzazione in base allo specifico contesto applicativo. I due casi d'uso più tipici nei quali è utile utilizzare la semantica bloccante dell'RPC sono i seguenti:

- quando un attore vuole inviare una sequenza di messaggi ordinata ad un certo destinatario, oppure quando vuole essere certo che il suo primo destinatario abbia ricevuto un messaggio prima di inviarne altri a destinatari diversi;

- quando lo stato del mittente varia in funzione della risposta ottenuta, rendendogli impossibile elaborare altre richieste finché non ottiene il risultato richiesto.

Occorre fare attenzione quando si adotta questo tipo di comunicazione, poichè un suo abuso, oltre ad influire negativamente sulle prestazioni del programma, può generare anche pericolose dipendenze tra gli attori, che potrebbero rimanere bloccati a causa di deadlock.

Vincoli di sincronizzazione locali

A causa del non determinismo di cui abbiamo già parlato precedentemente, quando i mittenti inviano i loro messaggi potrebbero non conoscere lo stato dei destinatari. In particolare, potrebbero non sapere se il destinatario si trova in uno stato che gli permetta di elaborare il messaggio, oppure se tale elaborazione al momento non è per lui possibile. Un primo modo per risolvere questo problema, consiste nel cosiddetto busy waiting: quando il destinatario non è in grado di processare la richiesta, la rifiuta; il mittente quindi è costretto a continuare a inviare richieste analoghe finché non vengono accettate, secondo il meccanismo chiamato polling. Questa soluzione però presenta due grossi svantaggi: il mittente non può proseguire con l'esecuzione del proprio programma, poiché si trova impegnato nell'inoltro continuo della stessa richiesta, quando invece potrebbe sfruttare il tempo in cui resta in attesa del risultato per eseguire ulteriore lavoro; inoltre viene generata una notevole quantità di traffico, che rischia di saturare ed intasare la rete del sistema. Per ovviare a questi problemi, si introduce un buffer, ovvero una struttura che permetta agli attori di memorizzare i messaggi ricevuti, ed eventualmente rimandare l'esecuzione, quando non sono in grado di portarla

a termine. Si contrasta in questo modo il non determinismo, che impedisce di conoscere l'ordine di arrivo dei messaggi con l'introduzione di vincoli che ne specificano invece un ordinamento parziale nell'elaborazione. Un'ultima questione riguarda come è possibile specificare l'ordinamento di esecuzione dei messaggi. Se ciò è fatto all'interno del codice dell'attore, si verifica una violazione del principio di separazione degli aspetti (separation of concerns), poiché vengono mischiate la logica delle funzionalità dell'attore, con quella di ordinamento dell'esecuzione dei messaggi. Una soluzione migliore è quella di avvalersi di predicati, che permettano di governare l'elaborazione di tali messaggi tramite espressioni logiche riguardanti lo stato dell'attore ed il tipo di messaggio ricevuto.

3.1.5 Pattern di programmazione parallela

Avendo già precedentemente introdotto alcuni pattern architetturali, è opportuno ora rilevare che alcuni presentano una forte affinità con il modello ad attori, in particolare il pattern pipeline ed il divide et impera.

Pipeline

Questo pattern modella le situazioni in cui è necessario eseguire un'elaborazione su dei dati, che viene suddivisa in più fasi o stage. Ogni fase può essere gestita come un componente a sé stante, permettendo così di riutilizzarla in elaborazioni diverse da quella corrente. Una volta eseguita questa suddivisione è quindi possibile costruire il processo di elaborazione necessario, chiamato pipeline, semplicemente combinando i componenti corretti in un particolare ordine. Un'applicazione classica di questo pattern, può essere trovata nell'elaborazione delle immagini. Questo pattern è particolarmente legato al modello ad attori, perché permette di sfruttarne il parallelismo in maniera semplice ed efficace. Nel caso di uno stream di immagini, realizzando ogni stage tramite un attore, è possibile eseguire contemporaneamente fasi diverse dell'elaborazione sui dati, aumentando notevolmente il throughput del sistema.

Divide et impera

Un altro pattern molto comune è il Divide et impera. Esso consiste in un approccio top-down ai problemi complessi, i quali in questo pattern, devono essere divisi in sottoproblemi più semplici da risolvere, eventualmente anche in maniera ricorsiva. Una volta raggiunto un livello di complessità sufficientemente ridotto, è possibile quindi risolvere i vari sottoproblemi generati,

ricombinando infine le soluzioni parziali, in modo da ricostruire la soluzione. A causa della sua grande genericità ed astrazione, talvolta il Divide et impera non viene nemmeno considerato un pattern architetturale, quanto piuttosto un approccio ai problemi, o un principio generale valido in molti altri ambiti, oltre a quello informatico. Lo stesso pattern pipeline può essere considerato un'applicazione di questo principio, poiché suddivide il problema iniziale lungo la dimensione dell'elaborazione. Applicando invece la suddivisione lungo la dimensione dei dati, che vengono quindi suddivisi, si arriva ad un altro pattern architetturale, il Map Reduce. Anch'esso trova una sua naturale realizzazione nel paradigma che stiamo studiando, poiché permette di modellare i singoli nodi computazionali, sia master che worker, proprio come attori.

Proprietà del modello ad attori

- Encapsulation: Riguarda la capacità degli attori di incapsulare lo stato e le comunicazioni, ciò gli permette di non condividere lo stato e di comunicare se non tramite messaggi.

- Fairness: il modello ad attori presuppone una nozione di equità per cui si afferma che ogni attore fa progressi se attività da svolgere, e che ogni messaggio è alla fine consegnato all'attore di destinazione, a meno che l'attore di destinazione non sia permanentemente indisponibile.

- Location Transparency: la posizione effettiva di un attore non influenza il suo nome. Attori comunicano scambiando messaggi con altri attori, che potrebbero essere sullo stesso Core, sulla stessa CPU, o su un altro nodo della rete.

- Mobility: La capacità di spostare gli attori su nodi diversi, che deriva dalla Location Trasparency.

3.2 Orleans

Come già detto all'inizio del capitolo, Orleans è un framework, che Microsoft sta tuttora sviluppando, grazie all'adozione di un modello di programmazione simile a quello ad attori, che permette ai programmatori di sviluppare le applicazioni Cloud più velocemente. Gli obbiettivi principali che si pone tale framework sono due:

- permettere anche agli sviluppatori che non hanno familiarità con i sistemi distribuiti, di realizzare applicazioni su larga scala;

- assicurarsi che tali applicazioni siano scalabili, senza che sia necessario riprogettarle e cambiarne l'architettura.

Per raggiungere questi obbiettivi il modello di programmazione di Orleans è stato ideato per aiutare gli sviluppatori a realizzare applicazioni in grado di scalare facilmente. Il sistema di programmazione di Orleans si basa su componenti asincroni, isolati e distribuiti, chiamati grain, che hanno molte caratteristiche in comune con gli attori. Però possono essere automaticamente replicati per ottenere la scalabilità; il loro stato può essere reso persistente su un sistema di storage condiviso, ed esistono meccanismi in grado di riunificare le modifiche dello stato, quando repliche diverse di uno stesso grain apportano ciascuna i propri aggiornamenti. In questa sezione verranno utilizzati esempi tratti dalla documentazione ufficiale di Orleans fornita da Microsoft, che riguardano l'implementazione di un social network molto semplice i cui utenti, modellati tramite la classe Chirper possono scegliere di seguire (follow) altri utenti, ricevendone i messaggi da questi pubblicati, come avviene per il noto social network Twitter.

3.2.1 Grain

I grain possono essere visti come l'unità base di programmazione: tutto il codice scritto per il framework Orleans viene infatti eseguito all'interno di un grain. Il sistema mantiene in esecuzione contemporaneamente più grain; essi non condividono memoria o porzioni di stato, sono internamente single-threaded, e portano a compimento l'esecuzione di ogni richiesta, prima di passare alla successiva, come succede anche per gli attori nel relativo modello. Le dimensioni dei grain non hanno limiti, tuttavia sta al progettista trovare una buona granularità per il proprio sistema, data dal compromesso tra la concorrenza e la quantità di stato, necessarie per garantire una computazione efficiente. Grain troppo grandi non permettono di sfruttare appieno il parallelismo dell'ambiente, costringendo computazioni che potrebbero essere eseguite concorrentemente ad essere invece eseguite sequenzialmente. Al contrario, grain troppo piccoli e numerosi porteranno ad un sovraccarico nelle comunicazioni necessarie per eseguire le richieste e per mantenere coerente lo stato interno del sistema, che risulterà eccessivamente frammentato. I grain comunicano tra loro tramite lo scambio di messaggi asincroni, che viene implementato tramite l'invocazione di metodi e che come per il modello ad attori, presenta una semantica non bloccante. L'eventuale valore di ritorno di tali metodi viene gestito da Orleans tramite il concetto di promise, o promessa, che rappresenta un risultato atteso nel futuro. Le promise sono il meccanismo che permettono di coordinare la computazione

concorrente del sistema. Ad esse può essere assegnata una porzione di codice, detta handler, che viene eseguita all'arrivo del risultato, mentre l'esecuzione del grain continua. Se invece è necessario si può decidere di adottare esplicitamente un comportamento bloccante, scegliendo di rimanere in attesa dell'arrivo di tale risultato. L'impredicibilità è una caratteristica intrinseca dei programmi realizzati tramite il framework Orleans, come d'altra parte lo è per ogni sistema distribuito sufficientemente complesso. Essa si esplicita nell'impossibilità di determinare l'ordine di esecuzione degli handler, poiché non è possibile conoscere anticipatamente con certezza il tempo necessario a trasmettere il messaggio attraverso la rete, né quello richiesto per l'esecuzione di ogni metodo invocato dal grain chiamante. Tuttavia, grazie al meccanismo delle promise, l'impredicibilità resta limitata a questo aspetto, e risulta quindi facile da gestire e comprendere, riducendo la possibilità che essa divenga una fonte di errori.

3.2.2 Activation

Come già detto all'inizio del capitolo, un grain può essere replicato automaticamente da Orleans per gestire aumenti nel carico di lavoro. Le singole istanze di un grain che sono in esecuzione nel sistema sono dette activation, o attivazioni. Esse sono in grado di elaborare richieste, indipendenti tra loro, che sono state rivolte ad uno stesso grain, eventualmente venendo anche eseguite su server diversi. Il processo di richiedere ulteriori macchine all'infrastruttura sottostante, viene quindi gestito automaticamente da Orleans, e tale infrastruttura viene completamente nascosta allo sviluppatore, che non deve in alcun modo farsene carico. Mentre i grain quindi sono l'astrazione logica a livello di programmazione, le attivazioni rappresentano l'unità di esecuzione a runtime. Ogni attivazione di uno stesso grain viene eseguita indipendentemente ed in maniera isolata dalle altre, ed esse non possono condividere memoria tra loro o invocarsi metodi reciprocamente. Esse sono quasi completamente trasparenti le une alle altre; la loro unica interazione, nascosta comunque allo sviluppatore, consiste nella fase di riconciliazione delle modifiche allo stato persistente del grain.

3.2.3 Promise

Il ciclo di vita di una promessa è abbastanza semplice, e comprende pochi stati. Inizialmente essa è detta unresolved e rappresenta, finché si trova in questo stato, l'attesa di ricevere un risultato in futuro. Quando tale risultato arriva, la promessa diviene fulfilled ed il suo valore diventa il risultato stesso. Se durante l'esecuzione della richiesta si è verificato invece qualche errore, la

promessa diventa broken e non ha valore. Sia che si trovi in stato di fulfilled che in stato di broken la promessa si dice resolved. Le promesse vengono implementate tramite due classi .NET:

- AsyncCompletion che rappresenta il futuro completamento di un'operazione
- AsyncValuehTi che rappresenta invece il futuro valore ottenuto come risultato di un'operazione

Gli handler associati alla risoluzione di una promessa sono invece implementati tramite delagati, passati come argomento al metodo ContinueWith() della promessa stessa. Tale metodo a sua volta restituisce una promessa. Se una determinata promise diventa broken, allora il delegato non viene eseguito, ed anche la promessa restituita dal suo metodo ContinueWith() diviene broken, a meno che non sia stato predisposto un delegato appositamente associato al fallimento della promise iniziale. Questo meccanismo permette di realizzare la propagazione degli errori, come permette di fare anche il meccanismo delle eccezioni, solitamente implementato dai linguaggi Object Oriented moderni. Le promesse mettono a disposizione anche i metodi Wait(), che sospendono l'esecuzione di un grain fino all'arrivo del risultato, e GetValue(), che forniscono il valore di tale risultato, bloccando l'esecuzione. L'esecuzione dei delegati all'interno delle activation è sempre single-threaded, quindi non può mai esserci più di un delegato in esecuzione all'interno di ogni singola attivazione. Segue il listato 3.1 che mostra un esempio di quanto detto finora, in particolare tramite il codice riportato viene creata una promessa, invocando un metodo su grainA, e ad essa viene associato un delegato invocando il metodo ContinueWith() sulla promessa stessa. Il delegato restituisce a sua volta una promise, sulla quale il grain corrente resta in attesa.

Listing 3.1: Creazione di una promessa, associazione di un delegato il quale a sua volta restituisce un'altra promessa, e comportamento bloccante specificato tramite l'invocazione del metodo Wait()

```
1 AsyncCompletion p1 = grainA.MethodA();
2 AsyncCompletion p2 = p1.ContinueWith(() =>
3 {
4  return grainB.MethodB();
5 });
6 p2.Wait();
```

3.2.4 Esecuzione di Activation

Quando un'attivazione riceve una richiesta, la esegue in unità di lavoro discrete, chiamate turni. Tutta l'esecuzione del codice di un grain, sia che si tratti di gestire un messaggio proveniente da un altro grain o da un client esterno, sia che si tratti dell'esecuzione di un delegato, avviene in un turno. Un turno arriva sempre alla sua conclusione, senza essere interrotto da altri turni della stessa attivazione. Nel complesso i sistemi Orleans possono eseguire più turni di attivazioni diversi in parallelo, ma ogni attivazione esegue i propri turni sequenzialmente; questo è il meccanismo che permette l'esecuzione di un activation single-threaded. A livello di scheduling tuttavia non c'è un thread dedicato per ogni attivazione, ma lo scheduler di Orleans si occupa di allocare di volta in volta i thread disponibili a turni di attivazione diverse. I thread vengono gestiti internamente come un pool. Grazie al modello di esecuzione single-threaded, non c'è bisogno di adottare strumenti per la sincronizzazione, come lock, semafori o monitor; per garantire l'assenza di fasi critiche tuttavia esso limita il parallelismo solo ad insiemi di grain ed attivazioni, impedendone l'utilizzo all'interno di uno stesso grain. Questa scelta è stata fatta per semplificare lo sviluppo di applicazioni, e ridurre notevolmente la probabilità di commettere errori, che solitamente si presentano numerosi quando gli sviluppatori si trovano a gestire questioni di sincronizzazione e concorrenza. Come già precedentemente affermato, esiste un certo grado di non determinismo nelle applicazioni sviluppate tramite Orleans, dovuto all'impredicibilità nella risoluzione delle promesse. Esso non genera comunque esecuzioni critiche, tuttavia richiede una certa attenzione, poiché occorre tener presente che lo stato di un'attivazione, quando un delegato viene eseguito, potrebbe essere diverso dallo stato che sussisteva quando il delegato è stato creato. Generalmente è necessario che un'attivazione finisca completamente di servire una richiesta prima di accettarne un'altra. In particolare, questo significa che non verranno accettate nuove richieste finché ci sono ancora promesse in stato unresolved, con relativi delegati non ancora eseguiti. Tuttavia è possibile intervenire per modificare questa politica in due maniere:

- marcando la classe del grain con l'attributo Reentrant, che permette ai turni di interferire liberamente;
- marcando i singoli metodi con l'attributo ReadOnly, che permette a tali metodi di interferire tra loro.

3.2.5 Persistenza dello stato

Ogni sistema Cloud generalmente è dotato di uno stato, che viene mantenuto persistente tramite funzionalità e servizi di storage. Questo avviene anche in Orleans: la persistenza è integrata nei grain che contengono parte dello stato dell'applicazione. Questo permette di rendere visibili e disponibili le modifiche che vengono effettuate da una specifica attivazione anche alle altre di uno stesso grain. Oltre alla porzione di stato persistente è tuttavia possibile avere una porzione temporanea, la quale viene mantenuta all'interno di una specifica attivazione, ed esiste solo durante il tempo di vita dell'attivazione stessa. Da segnalare che c'è la possibilità che un grain in un determinato momento esista solo in memoria persistente, se non ci sono richieste che lo riguardano; ciò significa che non esiste alcuna activation associata ad esso. Per ragioni di prestazioni Orleans può anche decidere di non riportare subito in memoria persistente le modifiche fatte allo stato di un grain, ma di mantenerle in memoria volatile, rendendole comunque disponibili a tutte le attivazioni. Questo comportamento aumenta la vulnerabilità del sistema: lo spegnimento delle macchine su cui queste modifiche non siano state memorizzate causa infatti la loro perdita.

3.2.6 Transazioni

Orleans fornisce un modello di consistenza basato su transazioni, che permettono di semplificare la gestione della concorrenza di un dato sistema. Tutti i grain che stanno processando una richiesta esterna vengono infatti inclusi in una transazione: ciò permette alla richiesta di essere eseguita atomicamente ed in maniera isolata. Le attivazioni che vengono eseguite all'interno di una data transazione, sono completamente isolate dalle attivazioni eseguite in altre transazioni e non possono accedere a dati modificati da transazioni non ancora completate. Le transazioni possono terminare con successo oppure fallire, e le loro modifiche vengono rese persistenti e visibili atomicamente solo in caso di successo, altrimenti vengono completamente annullate. Le transazioni implementano anche un semplice meccanismo per la gestione automatica degli errori: se un grain infatti fallisce o non è più disponibile, per un problema di rete o del server sul quale era in esecuzione, la transazione per la quale esso stava eseguendo del codice viene fatta fallire, e Orleans cerca automaticamente di rimandarla in esecuzione. In tal modo gli sviluppatori vengono liberati dalla responsabilità di gestire gli errori più comuni, legati a problemi hardware o software.

3.2.7 Meccanismi e politiche di scalabilità

Le tecniche più comuni per raggiungere la scalabilità sono l'asicronicità, il partizionamento e la replicazione. Essi vengono tutti inclusi nel modello di programmazione ed esecuzione di Orleans. I grain infatti si basano su meccanismi di comunicazione asincrona, e promuovono il partizionamento dello stato e della computazione. Inoltre Orleans può decidere a runtime di replicare le activation per aumentare le capacità del sistema, distribuire e bilanciare il carico di lavoro tra più server, automatizzando quindi la scalabilità. Tuttavia questi meccanismi non garantiscono automaticamente la soluzione di tutti i problemi di scalabilità. Essi devono essere infatti previsti all'interno dell'applicazione, scegliendo le corrette politiche, che non possono essere automaticamente adottate da Orleans, poiché sono strettamente legate alla semantica del problema. Ad esempio, se un grain diventa eccessivamente grande, la sua replicazione non può portare i benefici sperati, e sarebbe quindi necessario suddividerlo in più grain. Come ciò debba esser fatto tuttavia, dipende dalla semantica del grain che Orleans non può conoscere, quindi tale operazione non può essere fatta automaticamente. In sintesi Orleans fornisce i meccanismi, ma le politiche corrette per la loro utilizzazione devono essere introdotte da chi conosce la semantica dell'applicazione.

3.2.8 Interfacce dei grain

Piuttosto che sviluppare uno specifico linguaggio per la definizione di interfacce, Orleans utilizza le interfacce standard di .NET per definire quelle dei grain. Esse tuttavia devono seguire delle regole particolari:

- un'interfaccia di grain deve ereditare, direttamente o indirettamente, dall'interfaccia IGrain;
- tutti i getter[1] devono restituire una promise, mentre non devono esserci setter[2], poiché in ambiente .NET essi non hanno valore di ritorno;
- gli argomenti dei metodi devono essere interfacce di grain, oppure di un qualunque tipo serializzabile, in maniera tale da poter essere passati per valore.

[1] Nella programmazione ad oggetti la pratica di scrivere metodi di lettura e di scrittura degli attributi è talmente diffusa che in letteratura esistono nomi specifici per questo genere di metodi: getter (per un metodo che serve a leggere il valore di un attributo).

[2] setter (per un metodo che scrive un attributo)

3.2.9 Riferimenti ai grain

I riferimenti ai grain sono in realtà dei proxy, che implementano anch'essi l'interfaccia del grain a cui sono associati e ai quali forniscono l'accesso. Essi sono l'unico modo che ha un client per accedere ad un grain. Come per le promesse, anche i riferimenti ai grain possono trovarsi nei tre stati unresolved, fulfilled o broken. Il chiamante crea il riferimento ad un grain allocandone uno nuovo, oppure richiedendone uno specifico già esistente. Se tale riferimento è ancora in stato unresolved, e su di esso vengono invocati dei metodi, questi vengono automaticamente messi in coda in attesa di poter essere eseguiti, in maniera completamente trasparente allo sviluppatore.

3.2.10 Creazione ed uso dei grain

Per ogni interfaccia di grain Orleans genera automaticamente due classi:

- Una factory statica che permette di creare, eliminare o cercare determinati grain
- Un proxy, usato internamente dal framework per convertire le invocazioni di metodi in messaggi

La classe factory, nel caso più semplice, contiene i metodi per creare ed eliminare un grain, e per effettuarne il cast ad un riferimento di altro tipo; inoltre é possibile specificare delle annotazioni ai membri dell'interfaccia del grain, come ad esempio Queryable, per generare dei metodi aggiuntivi nella factory alla ricerca di grain che soddisfino determinate condizioni. Coerentemente con quanto affermato nella sezione precedente, invocando il metodo CreateGrain di una factory, viene immediatamente restituito il riferimento ad un grain, che può essere usato subito per invocare determinate operazioni su di esso. Queste verranno messe quindi in una coda fino al momento in cui la creazione verrà effettivamente completata. Se essa fallisce, tutte le promesse associate ai metodi invocati sul riferimento di quel grain falliranno automaticamente, come ci si aspetterebbe dai meccanismi di propagazione degli errori. Nel pezzo di codice 3.2 è mostrato un esempio di factory, in particolare quella necessaria alla gestione dei ChirperAccount. In essa compaiono i metodi necessari alla creazione, cancellazione e casting del grain, oltre a quello in grado di svolgere la ricerca basandosi sulla proprietà UserName, che è stata precedentemente dichiarata Queryable nell'interfaccia di tale grain.

Listing 3.2: Factory per la gestione del grain ChirperAccount: i primi due metodi si occupano della creazione e cancellazione di un ChirperAccount, il

terzo di effettuare il casting esplicito da un grain generico ad un ChirperAccount (necessario in caso di down-casting) ed il quarto in grado di effettuare la ricerca di un ChirperAccount all'interno del sistema, dato il suo UserName.

```
1  public class ChirperAccountFactory
2  {
3   public static IChirperAccount
4       CreateGrain(string name);
5   public static void
6       Delete(IChirperAccount grain);
7   public static IChirperAccount
8       Cast(IGrain grainRef);
9   public static IChirperAccount
10      LookupUserName(StringuserName);
11 }
```

3.2.11 Classi dei grain

Come già detto, la classe di un grain deve necessariamente implementare una o più interfacce di grain, ed ogni suo metodo deve restituire una promise. Tuttavia è possibile in fase di implementazione, specificare direttamente un valore di ritorno che non sia una promessa: esso verrà convertito a runtime in una promise da Orleans.

Ciò permette una maggiore semplicità e chiarezza nella scrittura ed analisi del codice. Come valore di ritorno è anche possibile specificare una promessa ottenuta dalla chiamata ad un altro grain, oppure mettere in scheduling l'esecuzione di un delegato. Quest'ultimo caso è quello mostrato nel listato 3.3.

Listing 3.3: Questo esempio mostra come sia possibile specificare al posto di un comune valore di ritorno l'esecuzione di un delegato.

```
1   FollowUser(string name)
2  {
3    IChirperPublisher user =
4          ChirperPublisherFactory.LookupUserName(name);
5
6     IChirperSubscriber me  = this.AsReference();
7
8     AsyncCompletion p = user.AddFollower(myName, me);
9      return p.ContinueWith(() =>
10      {
```

```
11          this.Subscriptions[name] = user;
12          });
13  }
```

3.2.12 Confronto con il modello ad attori

Come abbiamo potuto constatare analizzando il framework Orleans, esso adotta un modello molto simile a quello ad attori, introducendo però alcune differenze, con lo scopo di ottenere un vantaggio nella programmazione in ambienti Cloud. Queste differenze si concretizzano nella dissociazione presente tra il concetto di grain, che rappresenta l'attore in fase di programmazione, e quello di activation, a cui corrisponde invece la specifica istanza in esecuzione di un certo attore. Per quanto riguarda le quattro caratteristiche fondamentali del modello ad attori, possiamo affermare che sicuramente Orleans implementa l'incapsulamento e la trasparenza di locazione. Inoltre, nonostante non ci siano specifici riferimenti al concetto di fairness, possiamo supporre che la raffinata e complessa politica di scheduling adottata da Orleans, che deve, tra le altre cose, decidere anche quante activation assegnare ad un certo grain in ogni momento, sia in grado di garantire anche che nessun grain venga bloccato semplicemente perché il suo codice non viene più messo in esecuzione, da un certo momento in poi. L'unica proprietà che non viene quindi rispettata è quella di atomicità: questo discende da una precisa scelta, effettuata tenendo conto delle prestazioni del sistema, e dell'obbiettivo di raggiungere scalabilità automatica. Proprio per questo infatti viene introdotto il concetto di activation, e viene deciso che ad ogni grain possano esserne associate più di una. Voler rispettare l'atomicità sarebbe quindi assurdo, poiché implicherebbe che in ogni istante per ciascun grain, sia in esecuzione solo un activation. Per questo vengono infatti introdotte le complesse politiche di riconciliazione dello stato, che sono state solamente accennate nel corso di questa trattazione. Un'ulteriore opposizione alla proprietà di atomicità si trova nella possibilità di ogni activation, di servire concorrentemente più richieste, se i relativi metodi sono marcati come ReadOnly o la classe del grain come Reentrant. Ciò non comporta problemi, poiché queste marcature possono essere fatte solo se effettivamente l'esecuzione concorrente di più metodi non si traduce in esecuzioni critiche, con relativi problemi di consistenza dello stato.

3.3 Conclusioni

Come anticipato nell'introduzione di questo capitolo, l'analisi del modello ad attori e del framework Orleans dovrebbero averci fatto comprendere come essi siano molto più adatti alla realizzazione di applicazioni distribuite, in particolare applicazioni Cloud, di quanto non lo siano i modelli più tradizionali, come l'Object-Oriented. Affinchè questi nuovi modelli possano portare i benefici sperati, devono essere ben compresi e studiati da coloro che decidono di adottarli; occorre comprendere veramente il significato dei vari concetti offerti dai modelli e dei costrutti messi a disposizione dai linguaggi di programmazione che li implementano. Per comprendere questo, possiamo considerare l'analogia con il modello Object-Oriented. Esso ha permesso di realizzare applicazioni molto più strutturate, dotando i linguaggi di una semantica capace di supportare nativamente alcuni principi architetturali, come l'incapsulamento e l'uso di componenti, che altrimenti avrebbero dovuto essere innestati esplicitamente dagli sviluppatori. Tuttavia non vincola questi ultimi al rispetto ed all'adozione di tali principi, e non impedendisce di realizzare applicazioni mal progettate. Questo ci aiuta a capire che anche l'adozione del modello ad attori (o simili) non introduce automaticamente nel programma la capacità di sfruttare il parallelismo e, nel caso specifico di Orleans, la scalabilità. Esso permette di raggiungere tali capacità, a condizione che le entità fondamentali di tali modelli e linguaggi siano opportunamente utilizzate. Concludendo quindi, è importante oggi, per poter sviluppare applicazioni Cloud e distribuite, conoscere ed adottare nuovi paradigmi più adatti a tali ambienti d'esecuzione, ma anche conoscere i modelli, le architetture ed i pattern che sono in grado di sfruttarne le potenzialità e risolverne i problemi. Infatti chi per anni fosse stato abituato a realizzare programmi secondo un approccio sequenziale, e non avesse compreso i concetti fondamentali dell'Object-Oriented, probabilmente avrebbe continuato a lavorare secondo il suo metodo, realizzando applicazioni composte da un unico oggetto monolitico, nel cui metodo che funge da punto di ingresso del programma avrebbe trovato posto tutto il codice necessario alla sua esecuzione, realizzato come una lunga sequenza di istruzioni.

Capitolo 4

Caso reale

Introduzione

Come abbiamo potuto vedere in precedenza alla base del paradigma cloud, vi è una serie di concetti e tecnologie già esistenti e sviluppatesi separatamente. Le nuove evoluzioni tecnologiche, quali la banda larga hanno apportato un ulteriore grado di integrazione degli stessi.

Ad oggi l'Information Technology ha raggiunto una maturità tale da far sorgere un interesse crescente delle aziende, sulla possibile migrazione verso soluzioni Cloud, tali da permettere lo sfruttamento dei vantaggi offerti (pensiamo al self-service delle risorse o alla maggior flessibilità e scalabilità), consentendo nel contempo una possibile riduzione dei costi Infrastrutturali e di gestione.

Il cloud richiede di base un'idea di standardizzazione e uniformità, anche su centri di elaborazione remoti, per cui non è una soluzione che offre benefici per qualsiasi applicazione.

Prima di procedere con un progetto di migrazione che coinvolga elementi già esistenti, è necessaria un'analisi che permetta il confronto con le offerte disponibili sul mercato e stabilisca, a seconda dei diversi punti di vista affrontati, cosa possa essere migrato e cosa no.

Tale analisi prevede il confronto tra i workload dell'intero sistema informativo di origine e i diversi modelli di delivery cloud.

4.1 Regione Veneto

4.1.1 Descrizione

Questa realtà è costituita da un Infrastruttura a supporto di servizi ed applicazioni il tutto gestito da vari fornitori presso la sede della regione.

L'Infrastruttura si compone di una rete complessa nella quale ci sono Switch di livello 3 che gestiscono 61 Vlan 6 firewall di livello 9 e diversi gateway e router, che permettono l'accesso ai vari servizi/applicativi a utenti presenti sul territorio regionale/nazionale o agli uffici della regione presso la sede della comunità europea di Bruxelles.

Altri elementi presenti nell'Infrastruttura sono uno storage IBM DS8300 che eroga ai vari server le lun attraverso una rete SAN, a questo è collegato un SVC (System Storage SAN Volume Controller), che permette la duplicazione in tempo reale di parte dei dati su uno storage di Padova, attraverso un secondo SVC collegato a quello di Venezia da una rete ad alte prestazioni, garantendo l'alta disponibilità degli stessi necessaria per manenere in funzione il DR.

I backup vengono effettuati dal prodotto IBM Tivoli Storage Manager (TSM) in configurazione HACMP che fa uso di una tape library TL3684. Gli applicativi di controllo (Tivoli), le applicazion di backup (TSM), il sistema di service Desk e Asset Management (IBM TivoliMaximo) e i database Oracle 9i e 11g girano su lpar messe a disposizione da due IBM P570 che garantiscono anche la ridondanza hardware necessaria ai cluster Oracle e IBM HACM.

Il resto dei server di collaudo e parte delle macchine di produzione, compresi i server di posta girano su dei virtual center vmware ospitati su blade IBM HS22. Rimangono alcuni server che per vari motivi non sono stati virtualizzati, e quindi sono su vecchi sistemi blade HP o nuovi Blade IBM. I server applicativi divisi tra server di front-end e application server permettono di erogare un totale di 81 applicazioni.

Segue una tabella riassuntiva che da un dettaglio sulle applicazioni presenti sui vari server.

Nome	Categ.	Tec.	Tec. Base	AS	DB
Gestione accessi	Infrastr.	CS-W	FORMS	forms6i	Ora 9i
Menu SIRV	Infrastr.	CS-W	FORMS	forms6i	Ora 9i
Progile Manager	Infrastr.	WEB	Java	Jboss	Ora 9i
PROCOM Programmi Comunitari	MC	WEB	FORMS	forms6i	Ora 9i
PROCOM 2007	MC	WEB	FORMS	forms6i	Ora 9i
PROSEDIT-Gestione	BF	CS	Delphi		Ora 9i
PROSEDIT-Interrogazione via web	PO	WEB	Java	Jboss	Ora 9i
IRA -Web	BF	WEB	FORMS	forms6i	Ora 11g
Istruttoria PSR 2007 -2013	BF	WEB	Java		
Presentazione PSR 2007-2013	BF	WEB	Java		
Indirizzario	MC	CS-W	FORMS	forms6i	Ora 9i
Anagrafe ditte settore primario	MC	WEB	Java	Jboss	Ora 11g
Modello unico di domanda	MC	WEB	Java	Jboss	Ora 11g
Nomine	BF	WEB	Java	Jboss	Ora 11g
Finanziario2000	BF	CS	Pbuild		Ora 9i
Alimento versioni finanziario 2000	Infrastr.	CS	Pbuild		Ora 9i
Finanziario 2000 euro	MC	CS	Pbuild		Ora 9i
F2K MUTUI	MC	CS	Pbuild		Ora 9i
Budgetweb	MC	WEB	Java+Pbuild	Jboss	Ora 9i
Nuovo Geac	MC	CS	Pbuild		Ora 9i
BIBICO	BI	CS	Pbuild		Ora 9i
Buoni Scuola	BF	CS	Delphi		Ora 9i
Collaudatori Regionali	BF	CS	FORMS		Ora 9i
Segnalazioni MAC/Mev	BF	WEB	Java	Jboss	Ora 9i
Gestione atti ispettivi	BI	CS	Delphi		Ora 11g
VRU-Virtualizzazione Risorse Umane	BI	WEB	Java	Jboss	Ora 9i
Patenti Fitosanitari	BI	WEB	Delphi/Java	Jboss	Ora 11g
Modulo Trasparenza	MC	WEB	Java	Jboss	Ora 11g
RVT-WEB	MC	WEB	Java	Jboss	Ora 9i
MT-WEB	MC	WEB	Java	Jboss	Ora 9i
Certificati Fito Sanitari	MC	WEB	Java	Jboss	Ora 9i
Contributi Trasporto Pubblico	BF	CS		Delphi	Ora 9i
Gest. Patrimonio-Inventario Beni mobili pat. web	BI	WEB	Java	Jboss	Ora 9i
MOMA	Infrastr.	WEB	Java	Jboss	Ora 9i
Concessionari WEB	PO	WEB	Java	Jboss	Ora 9i
Mail Dispatcher	Infrastr.	WEB	Java	Jboss	Ora 9i
CAS (Central Autentication Service)	Infrastr.	WEB	Java	Jboss	Ora 9i
Gusi-MISS	MC	WEB	Java	Jboss	Ora 11g
Gusi-RISS	MC	WEB	Java	Jboss	Ora 11g
SAPER-FISCALDATA	BF	WEB	Java	Jboss	PostgreSql
Monitoraggio Direttiva Nitrati	MC	WEB	Java	Jboss	Ora 11g
PMC					
Bonifica Anagrafica Bollo Auto	BF	CS-W	Delphi		PostgreSql
CO.RE.COM.	BF	CS-W	Delphi/Java		PostgreSql
Certificazioni Forestali PEFC	BI	WEB	Java	Jboss	Ora 11g
FIRMAWEB	Infrastr.	WEB	Java	Jboss	Ora 9i
Catalogo Aiuti Sett. Agricolo-CAR	BF	N/D	Java	Jboss	Ora 11g
AGRITURISMO	BF	WEB	Java	Jboss	Ora 11g
EQUINI	BF	WEB	Java	Jboss	Ora 11g
SMUPR Monitoraggio Progetti	BI	WEB	JAVA	Jboss	Ora 9i
InterOP CSM DB-Portale Cresci	MC	WEB	PHP/Joomla	Apache	Mysql
OverNetwork	PO	WEB	PHP/Joomla	Apache	Mysql
SVIR-Interop-Cresci	MC	WEB	Java		

Nome	**Categ.**	**Tec.**	**Tec. Base**	**AS**	**DB**
BIRV	Infrastr.	WEB	Java	Jboss	Ora 9i
DIRV Dorsale integrazione Regione Veneto	Infrastr.	WEB	Java	Spagic	Ora 9i
UMA-Uffici Motorizzazione Agricola	BF	CS-W	Delphi		Ora 11g
w UMA-Uffici Motorizzazione Agricola	BF	WEB	Java	Jboss	Ora 11g
Finanziario 2000-Poli Museali	BF	CS	PBuild		Ora 9i
DBMap ABACO	Infrastr.	N/D			
Criptoserver	MC	WEB	DOTNET		
Riscossioni Coattive	BF	Pbuild			
Telefonia WEB	BF	N/D	Java		
MONIT INTEROP	MC	WEB	Java	Jboss	Ora 11g
Gestione LR 11-2009	BF	WEB	Java	Jboss	Ora 11g
Anagrafe contabile web	BF	WEB		Jboss	Ora 9i
F2k Scuola Polizia Locale	BI	CS			Ora 9i
Certificato Fito Sanitari GUCF	MC	WEB			
Siti Patri ABACO	BF	WEB	Java+C/S	Jboss	Ora 9i+Ora-Spatial
Sportello Unico Richieste	BI		Java	Jboss	Ora 11g
Gestione Richieste	BI		Java	Jboss	Ora 11g
IPCC Monitoraggio Direttiva Nitrati	MC	WEB	Java	Jboss	Ora 11g
SMUPR Sender	BI	WEB			
Abilitazione Vigilanza CAA	BF	WEB	Java	Jboss	Ora 11g
LR12 Qualità Verificata	BF	WEB	Java	Jboss	Ora 11g
Fattorie Didattiche	BF	WEB	Java	Jboss	Ora 11g
Agri Quality Bank-Vigilanza qualità cert.	BF		Liferay	Tomcat	Ora 11g
Business Plan	BI	WEB	Java	Jboss	Ora 11g
Banda Larga	PO	WEB	PHP/Joomla	Apache	Mysql
NICA	Infrastr.	PRODOTTO	Java		
CresiWeb	MC	WEB	Tomcat		Ora 9i
Automezzi Web	BF	CS-W	PHP		
Mon. Formazione-Iscrizione su portale	BF	WEB	Java	Jboss	Ora 9i

4.1.2 Soluzione in Cloud Computing

Per poter meglio comprendere le scelte proposte in questo caso pratico, comincio citando quanto riportato in un lavoro eseguito da DigitPA dal titolo 'Raccomandazioni e proposte sull'utilizzo del cloud computing nella pubblica amministrazione'.

In termini generali, le pubbliche amministrazioni devono avvicinarsi ai servizi cloud privilegiando atteggiamenti orientati alla prudenza e alla consapevolezza come suggerito, tra gli altri, dal Garante per la protezione dei personali ed ENISA. Entrambe le istituzioni raccomandano un'attenta valutazione dei rischi legati alla fruizione di servizi IT in modalità cloud computing al fine di preservare confidenzialità e integrità dei dati dei cittadini, l'integrità e la continuità dei servizi loro offerti, il loro diritto alla privacy ed infine e più in generale l'interesse e la sicurezza nazionale. L'adozione di servizi cloud ha una grande attrattiva per i potenziali CSC[1] *per i vantaggi in termini di costo, flessibilità, elasticità e agilità, ma nel contempo, suscita anche preoccupazioni, legate a rischi, in alcuni casi reali o in altri solo percepiti, riguardo la capacità dei CSP*[2] *di offrire adeguati livelli di protezione dati e delle applicazioni, controllo, affidabilità, trasparenza e conformità legale.*[23]

Per quanto visto finora la soluzione pubblica è esclusa, perchè pur essendoci una notevole flessibilità e scalabilità, con una conseguente riduzione dei costi, l'impossibilità di avere un controllo sull'Infrastruttura e soprattutto sulle scelte riguardanti le policy di sicurezza, mette in discussione la sicurezza dei dati da attacchi esterni e/o interni.

Questo ci porta a valutare una soluzione Privata o Comunitaria; se si vuole aver un controllo ulteriore dell'ubicazione dei dati più sensibili, è possibile pensare ad una soluzione di cloud Privato o Comunitario on-permise.[19]

[1] Il cloud consumer (o cloud service consumer, CSC) è il principale soggetto che utilizza i servizi di cloud computing. Un cloud consumer rappresenta una persona o una organizzazione che ha sottoscritto un contratto con un cloud provider. Il cloud consumer esamina il catalogo dei servizi di un cloud provider, richiede specifici servizi e li utilizza. Il cloud consumer utilizza degli accordi sui livelli di servizio (Service Level Agreements / SLAs) per specificare i requisiti sulle prestazioni tecniche che devono essere soddisfatti da un cloud provider. Un cloud provider potrebbe anche elencare negli SLA un insieme di restrizioni e obblighi che il cloud consumer deve accettare.

[2] Il cloud provider (o cloud service provider, CSP) è il soggetto responsabile di rendere il servizio utilizzabile alle parti terze interessate. Il cloud provider acquisisce e gestisce le infrastrutture elaborative necessarie a fornire i servizi, assicura l'esecuzione dei programmi che consentono i servizi, e le infrastrutture per erogare i servizi attraverso la rete.

La soluzione ottenibile con un cloud Privato fig. 4.1 on-permise implica la realizzazione di un ambiente informatico interno all'azienda stessa, realizzato con la virtualizzazione delle risorse, dei servizi e standardizzandone la gestione.

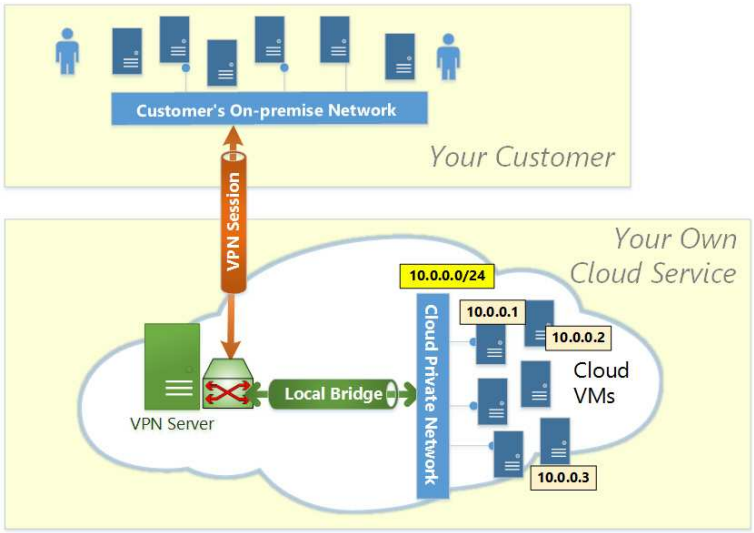

Figura 4.1: Cloud Privato on permise network

Esistono anche soluzioni intermedie ad esempio, Amazon Web Service (AWS) mette a disposizione VPC[3] fig. 4.2 viene considerato pubblico perchè usa le risorse di calcolo comuni, rese disponibili da parte di Amazon. Tuttavia è anche privata per due ragioni: in primo luogo il collegamento tra IT legacy e il Cloud è garantito da una rete privata virtuale, che sfrutta il vantaggio di sicurezza del Cloud privato. Tutte le politiche di sicurezza aziendale si applicano ancora alle risorse sul Cloud, anche se si trovano su un Cloud pubblico. In secondo luogo, AWS dedica una serie di risorse "isolate" per la VPC. Questo non significa che gli utenti siano costretti a pagare queste risorse "isolate"; essi infatti possono continuare a usufruire del "pay-per-use". VPC rappresenta un perfetto equilibrio tra controllo (Private Cloud) e flessibilità (Public Cloud). Tale soluzione però non permette di scegliere l'ubicazione fisica dei dati in Cloud, per questa ragione la migrazione in questo caso si pone due obbiettivi :

1. la suddivisione dei componenti tra quelli da migrare in cloud e quelli da mantenere al di fuori.

[3]Virtual Private Cloud un "ponte" sicuro e senza soluzione di continuità tra un'organizzazione di Infrastruttura IT esistente e il Cloud Amazon pubblico.

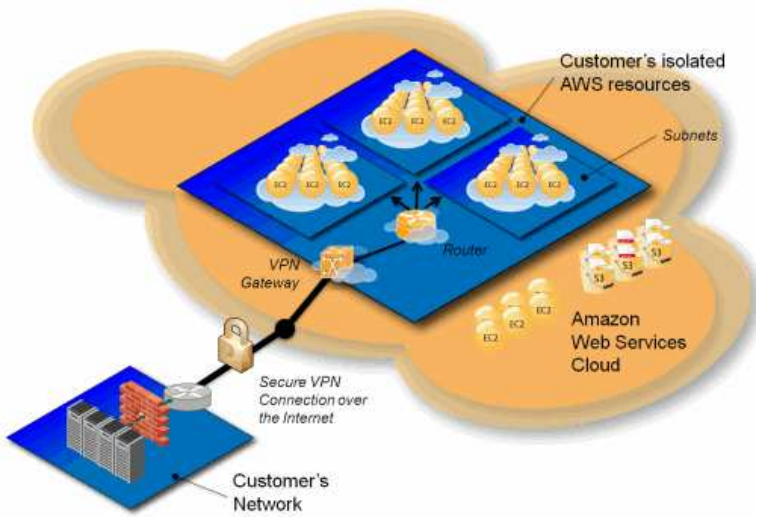

Figura 4.2: Virtual Private Cloud

2. la parte di controllo di accessi e permessi, quindi la migrazione di Access Control List (ACL) così da garantire il mantenimento del livello di sicurezza attuale.

I fattori presi in considerazione per il punto uno sono quindi i tempi di risposta (cercando di mantenerli più bassi possibili), la data privacy (per cui i dati sensibili vanno mantenuti on-premise.[4]

Si cerca infine di avere un risparmio sui costi il più alto possibile, modellando sia i benefici portati dalla migrazione che i nuovi costi, siano essi effettivi (per esempio un aumento dei costi di comunicazione) oppure derivati (per esempio l'aumento dei tempi di risposta nelle transazioni).

La soluzioni percorribili in queste condizioni sono soluzioni on-permise, nella quale si utilizzera il modello di tipo IaaS/Privato, o VPC nella quale i database e lo storage rimangono nel ced di regione Veneto, mentre tutti gli altri server diventano oggetto di valutazione, se rimanere on-permise o passare in cloud.

Per ottenere un maggior risparmio si potrebbe pensare, unendo piu enti pubblici, all'utilizzo di un modello sempre on-permise di tipo IaaS/Comunitaria, questo introdurrebbe ulteriori possibilità di risparmio mantenendo missione, requisiti di sicurezza, vincoli di condotta e di conformità adeguati.

A livello comunitario il cloud computing è ormai considerato come una strategia da perseguire, e ciò è riflesso anche nei programmi dell'Agenzia

[4]On-premises software (sometimes abbreviated as on-prem software) is installed and run on computers on the premises (in the building) of the person or organisation using the software

per l'Italia Digitale (AgID)[25], che in collaborazione con la CONSIP[29], ha completato alla fine del 2014 la Gara per i servizi Cloud nella PA.

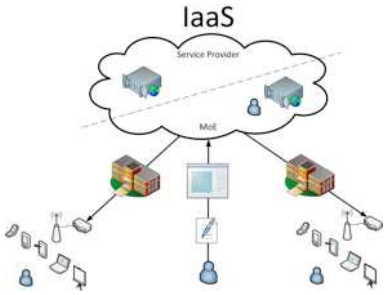

Figura 4.3: IaaS

La maggior parte delle infrastrutture di cloud computing IaaS utilizzano i più comuni hypervisor: VMware, Hyper-V, XEN e KVM. Le singole aziende decidono se tariffare il cliente con un pagamento mensile o orario, ma in tutti casi viene fornita la possibilità di aumentare o diminuire le risorse acquistate; non tutti gli attuali fornitori presenti nel nostro paese, forniscono un set di API per interagire con il cloud, il che rende questi sistemi ancora del tutto paragonabili al modello di Amazon EC2, ma sicuramente utilizzabili per startup, PMI e grandi aziende.

Nello studio *Workload migration into clouds*[18]; si sostiene come i due aspetti che più influenzano la valutazione della migrazione siano costi e rischi connessi. La migrazione viene vista come un percorso completo che consiste in sei fasi (fig. 4.4):

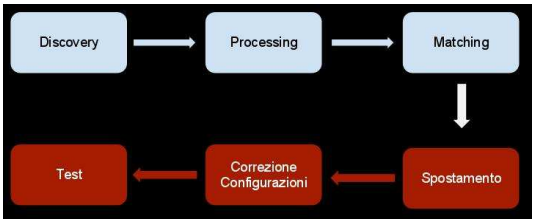

Figura 4.4: Fasi migrazione

- discovery, nella quale vengono esplorati i sistemi originari tramite tool appositi;
- processing (analysis e design), nella quale si selezionano i workload destinati alla migrazione e si valutano anche i costi, cercando un Return

of Investment (ROI) positivo. A tal proposito i vari provider forniscono dei tool fruibili liberamente, per facilitare tale attività[5] fig. 4.5.

- matching, in cui si prepara un file che descrive precisamente le origini e le destinazioni e che verranno utilizzate nella successiva fase
- spostamento, vengono create le immagini di destinazione
- configurazione, si configurano le immagini di destinazione e si apportano eventuali correzioni.
- test, si testa il risultato finale.

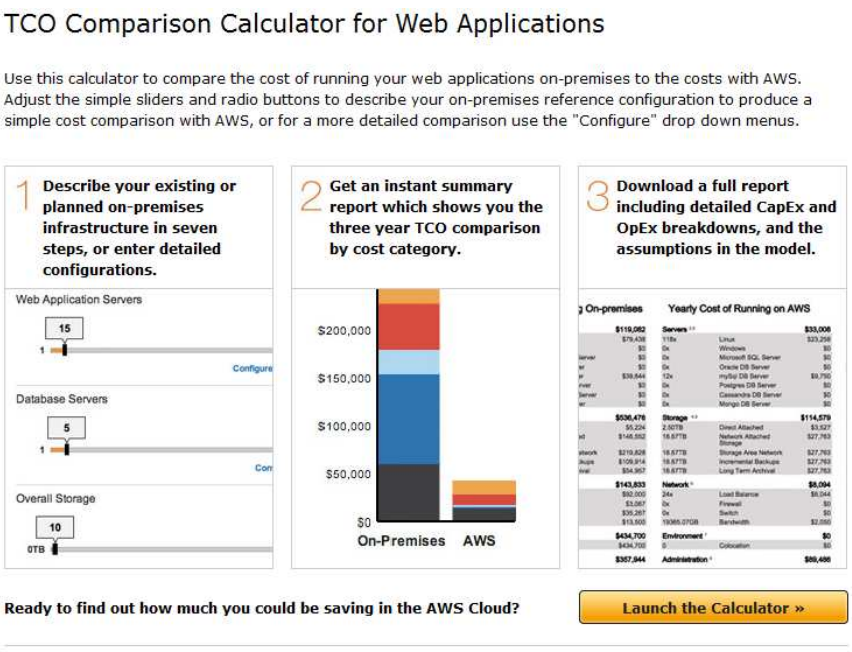

Figura 4.5: Comparazione costi totali di possesso

Nel caso in cui vi sia un elevato numero di piattaforme di destinazione, gli autori propongono un framework di nome Darwin[21], che contiene al suo interno vari componenti dedicati alle singole fasi e che sono anche utilizzabili separatamente.

[5] http://azure.microsoft.com/it-it/pricing/calculator, http://roitco.vmware.com, http://asw.amazon.com/tco-calculator/, http://cloud.it/cloud-computing/calcola-costo-cloud.aspx, http://www.rackspace.com/cloud/comparison, https://www.rightscale.com/cloud-analytics-free-trial

Concludendo, vengono segnalati una serie di problemi riscontrati nella fase di test, fra i quali il problema della mancanza di API standard sul cloud, mancanza di strumenti integrati per il calcolo del ROI, e limiti alla configurabilità dell'ambiente di destinazione, sui quali gli autori intendono lavorare in futuro.

Ritornado ad analizzare il caso particolare di Regione Veneto il totale delle macchine non è elevato, e i sistemi operativi che vengono utilizzati per far funzionare i vari server sono già virtualizzati e appartengono alla lista sotto riportata :

Windows 2003
Windows 2008 sp2
Linux Redhat 6.5
Linux RedHat 5.10
AIX 5.3
AIX 7.1

La fase di discovering risulta molto agevole, in quanto molte delle macchine sono già in un ambiente virtualizzato vmware, e si può usare un prodotto free RVTools, che permette di estrarre le configurazioni di tutte le macchine virtuali presenti in un virtual center. Il virtual center vmware mette a disposizione degli strumenti, che permettono di valutare il carico e l'utilizzo delle risorse per le singole macchine.

Questo permette l'estrazione e la valutazione delle risorse necessarie al dimensionamento delle macchine fornite per la migrazione al cload. In una fase sucessiva viene prevista un analisi del'uso delle risorse di rete per dimensionare la parte gestita dal cloud sulle macchine oggetto della migrazione.

Possono essere valutati i costi in relazione alle scelte possibili legate al workload desiderato. Nel nostro caso pratico, avvalendosi del cloud per i server di frontend, e sfruttando la possibilità di allocare e disallocare in modo dinamico i server in caso di sovraccarico, si vede subito un vantaggio sui tempi di accesso al servizio che al momento risultano critici in alcune fasce orarie.

In questo particolare ambiente è previsto un disater recovery, che prende un particolare significato in un ottica di Cloud computing; di fatto al momento assorbe dei costi per garantire la continuità, in quanto è previsto uno storage e delle macchine pronte a garantire i servizi minimi.

Il cloud computing offre un'alternativa interessante ai modelli tradizionali di disaster recovery. Il "cloud" è intrinsecamente un'Infrastruttura condivisa: un insieme di risorse, con costi Infrastrutturali distribuiti tra tutti coloro che stipulano contratti per utilizzare queste risorse come servizio cloud.

Questa natura condivisa rende il cloud computing un modello ideale per i processi di disaster recovery. Anche estendendo la definizione di disaster recovery a interruzioni più banali e frequenti dei servizi, l'esigenza di queste risorse di ripristino risulta sporadica. Poiché è estremamente improbabile che tutte le organizzazioni che si affidano al cloud computing per i processi di backup e ripristino, abbiano bisogno dell'Infrastruttura nello stesso momento, è possibile ridurre i costi e i tempi di ripristino.

I servizi gestiti di business resiliency basati su cloud, come IBM SmartCloud Virtualised Server Recovery, sono stati ideati per raggiungere un equilibrio tra la convenienza economica del ripristino su un'Infrastruttura fisica condivisa e la rapidità dell'Infrastruttura dedicata. Poiché le immagini dei server e i dati vengono replicati continuamente, il tempo necessario per ripristinarli può essere ridotto drasticamente a meno di un'ora, e in molti casi a pochi minuti, o perfino secondi.

Tuttavia, i costi sono più simili a quelli di un'Infrastrastruttura condivisa. Sebbene il cloud computing offra molti vantaggi come piattaforma di disaster recovery, è necessario considerare alcuni aspetti essenziali nella pianificazione della transizione ad una strategia di business resiliency basata sul cloud computing, e nella scelta del fornitore di servizi cloud. Questi aspetti comprendono:

- Accesso al servizio tramite un portale con funzionalità di failover e failback.
- Supporto per il test dei processi di disaster recovery.
- Più livelli di servizio.
- Supporto per ambienti server eterogenei e virtualizzati.
- Capacità di operare in tutto il mondo e presenza locale.
- Migrazione da e coesistenza con metodi tradizionali di disaster recovery.

Questi aspetti vengono esaminati più dettagliatamente nelle sezioni seguenti.

4.1.3 Vantaggi

Come abbiamo visto il Cloud computing è in grado offrire diverse caratteristiche di servizio che risultano utili per l'user finale.

Infrastruttura "agile": i servizi IT, basati su Cloud, sono sempre più in grado di sostenere una forza lavoro crescente, permettendo ai consumatori di poter accedere alle proprie risorse ovunque essi si trovino.

- Risposte rapide: il Cloud permette di potersi espandere facilmente, potendo soddisfare le esigenze. Non è più necessario alle aziende dover acquistare risorse di calcolo, in quanto il Cloud garantisce variazioni e richieste "on-demand" della potenza di elaborazione e di calcolo.

- Minor costi: non si hanno più costi eccessivi di manutenzione delle infrastrutture IT. Un servizio Cloud è disponibile "on-demand", seguendo la logica del "pay-per-use", ovvero acquistando solo quello di cui si usufruisce. Si riducono sprechi di risorse, e di conseguenza la spesa. Se si necessita di una maggior potenza di calcolo, è possibile adottare nuovi applicativi senza dover affrontare investimenti onerosi.

- Minor guasti al sistema: con l'adozione del Cloud computing, si evitano problemi di interruzione del servizio. Ciò significa, che se si verifica un guasto ad un singolo nodo, si continua ad avere comunque funzionalità dal servizio. I tempi di ordinazione, realizzazione, installazione e configurazione sono estremamente ridotti. Così come i costi di manutenzione e riparazione.

- User friendly: la sua interfaccia risulta alquanto semplice, e nella maggioranza dei casi è Web; porta ad un facile rapporto tra l'user e l'applicazione Cloud in sè.

4.1.4 Svantaggi

Sicuramente uno dei problemi principali a cui si deve far fronte, riguarda la privacy e la sicurezza dei propri dati.

Questa perplessità riguarda la "riservatezza ed la legalità", non solo dei dati personali, ma ancora di più dei dati sensibili. L'uso di servizi Cloud comporta l'accettazione di termini d'uso, e i dati vengono trattati secondo i termini acettati. Altri problemi riguardanti il Cloud, nella maggior parte dei casi, possono derivare dalla gestione della "rete". Con l'accettazione di una soluzione Cloud la rete diviene una risorsa critica: la velocità, il rischio di perdere pacchetti o dati nel trasferimento, ecc. Può capitare che si arrivi anche ad una saturazione di accesso al servizio, dovuta alle tante richieste, con conseguenti problemi di accesso.

Una cattiva gestione delle risorse può portare ad un aumento dei costi, derivanti dall'incapacità di far spegnere e quindi risparmiare risorse allocate e non utilizzatre.

4.1.5 Conclusioni

Per questa realtà ci sono dei limiti come anticipato sull'ubicazione dei dati, che devono essere mantenuti in locale, ne consegue una minore gamma di possibilità nella scelta di una soluzione cloud. La proposta di una soluzione Intermedia dove si migrano i frontend e i backend, mantenendo i server su cui girano i db in locale per diminuire i tempi di accesso allo storage, vincola anche la possibilità di un ritorno economico sostanzioso. Andrebbe fatta un'analisi più dettagliata sulle possibili alternative, e magari una misura mirata ai tempi di accesso. Una soluzione Cloud introduce la possibilità di usufruire delle risorse in maniera più dinamica, ottimizzando le prestazioni deì vari servizi, andando incontro alle nuove direttive comunitarie sulla pubblicha amministrazione.

Capitolo 5

Conclusioni

Il cloud computing è da anni un argomento di discussione e analisi nel mondo accademico e dell'Information Technology. Sono state date definizioni che con l'evoluzione tecnologica degli strumenti a disposizione dei Provider si sono ampliate e consolidate. La sua natura di servizio non chiaramente localizzato ha generato diverse problematiche, non solo a livello tecnologico ma anche a livello legale (dei problemi legali non ci siamo occupati specificamente, ma li citiamo in quanto causa di frizioni nell'adozione di tale tecnologia). Per esempio all'interno della comunità economica, al fine di stimolare l'adozione del cloud computing, si sta promuovendo un processo per uniformare la normativa sulla proprietà dei dati. Alcune grosse aziende come CISCO, per ovviare alla inadeguatezza della normativa, inseriscono nel contratto di cloud clausule dove si assumono i rischi derivanti dalle diverse normative presenti ove i dati vengono collocati. Partendo dalla definizione, analizzando l'implementazione nonchè l'evoluzione degli strumenti a disposizione di chi programma in questo ambito, ci si rende conto quanto questo paradigma sia in evoluzione, e quanto ancora possa crescere. Si sta facendo un grande lavoro sia sulla virtualizzazione delle reti, sia sulle tecnologie di storage, che in futuro permetteranno con la diffusione della banda larga, di superare determinati limiti ad oggi presenti in determinati ambiti del Cloud, favorendone la diffusione e l'utilizzo. Dallo sviluppo e analisi svolta, si evidenzia come il paradigma cloud sia altamente modulabile rispetto alle esigenze aziendali proprio per le numerose opzioni e soluzioni tecnologiche, software e gestionali. Di seguito si riassumono gli elementi di vantaggio/svantaggio rilevabili nel paradigma cloud.

vantaggi nel passaggio a cloud: La gestione dell'infrastruttura è messa a disposizione dal Cloud Provider, ne deriva che anche la sicurezza e affidabilità dei Data Center è a carico suo. Permette di passare da CAPEX a OPEX; il pagamento avviene in base all'utilizzo. Vi è una minor necessità di

assumere personale specializzato che si occupi della gestione dell'infrastruttura. In molti casi le aziende che forniscono la soluzione cloud prevedono la possibilità di mettersi al sicuro da problemi che potrebbero accadere al Data Center. Il cloud computing permette di reagire rapidamente a esigenze di carico improvviso. Lo scaling è anche possibile verso il basso. La larga maggioranza dei servizi cloud sono forniti con un interfaccia Web. Riduzione dei rischi di impresa e dei costi operativi e di manutenzione. Un servizio a costi inferiori, inoltre è agevolata riorganizzazione interna dell'azienda.

svantaggi nel passaggio al cloud: esistono ancora delle problematiche aperte nell'ambito della ricerca. Problemi di connettività possono provocare un blocco completo del processo produttivo legato alla disponibilità dei sistemi informativi cloud. In Italia la mancanza di una banda larga e affidabile, distribuita su tutto il territorio, rende spesso difficile per molte aziende localizzate al di fuori dei grandi centri urbani l'accesso a servizi di tipo cloud; Pericolo di perdita di controllo, per esposizione dei dati sensibilio al Cloud. Impossibilità di mantenere certificazioni acquisite. Indisponibilità del provider per audit di terze parti, indisponibilità del provider ad acquisire certificazioni. Mancanza di requisiti per specifiche certificazioni. La cancellazione dei dati non implica l'immediata cancellazione fisica. Scarsa preparazione a gestire il modello Cloud. Non sempre è possibile un integrazione con gli apparati locali.

Appendice A

Appendici

A.1 Link usati nella raccolta delle informazioni

N	Nome	Categoria	Link
1	Open Virtualization Alleance	Comunity	OVA
2	KVM	OpenSoftware	KVM
3	lxc,lxd,lxcfs,cgmanager	OpenSoftware	Linux Containers
4	Openvz	OpenSoftware	OVZ
5	WPAR	Prodotto Commerciale	Work Load Partition
6	Solaris Zone	Prodotto Commerciale	Solaris Zone
7	linux-vserver	OpenSoftware	Linux Vserver
8	Cloud Foundry	OpenSoftware	Cloud Foudary org
9	Openshift	Prodotto Commerciale	Open Shift
10	Dokers	OpenSoftware	Dockers
11	Mesos	OpenSoftware	Mesos
12	Coreos	OpenSoftware	Coreos
13	Developer network MSD	Archivio	Archivio MSD
14	Code Sample MSD	Archivio	Codice di esempio MSD
15	Bluemix	Provider	Cloud Provider IBM
16	OpenNebula	Community	Open Nebula
17	OpenStack	Comunity	Open Stack
18	Open switch	OpenSoftware	Open Switch
18	Juju	OpenSoftware	Juju
20	MAAS	OpenSoftware	Metal as a service
21	Open Compute	Open project	Open Compute
22	Contributo a openstack	statistiche	Statistiche OpenStack
23	Cumulus Linux	Open project	Cumulus Linux
24	occi-wg	Community	occi-wg
25	CloudBase	Provider	Cloud Base
26	RDO	OpenProject	RDO
27	Nist Fraimwork Definition	Figura	Nist
28	Caratteristiche essenziali	Figura	Caratteristiche
29	Classi di distribuzione	Figura	Classi
30	Modelli di servizio	Figura	Modelli Servizio
31	Modelli di distribuzione	Figura	Modelli distribuzione
32	Cloud Privato on permise network	Figura	On permise
33	Virtual Private Cloud	Figura	vpc
34	IaaS	Figura	IAAS
35	Comparazione costi totali di possesso	Figura	Costi

N	Nome	Categoria	Link
36	Hyper-V	Software	hyper-v
37	Xen	Software	Xenhyp
38	AWS	Provider	AWS
39	Rackspace	Provider	RackSpace
40	Vmware	Software	VMWARE
41	Virtualbox	Software	Virtualbox
42	Parallels	Spftware	Parallels
43	Virtualization Technology	Hardware	VirtualizationTechnology
44	Eucalyptus	Software	eucalyptus
45	dmtf	Comunity	dmtf
46	Classificazione Hypervisor	Figura	Classificazione Hypervisor
47	Aeolus	Opensource Project	Aeolus Project
48	Bonfire	Open Project	Bonfire
49	Cirrus	Open Project	Cirrus
50	FutureSystems	Open Project	FutureSystems
51	OSv	Open Project	OSV
52	Openstack Module	Figura	Openstack Modules
53	Eucalyptus	Software	Eucalyptus
54	Ganglia	Software	Ganglia
55	Aeolus	Project	Aeolus
56	Occiware	Project	Occiware
57	Orleans Microsoft	Software	Orleans Download
57	Orleans	Software	Orleans
57	Orleans	Software	Orleans link
58	Cloud computing analisi dei modelli architetturali	Tesi	Analisi dei modelli architetturali
59	Cloud Computing Marcello Zin	Tesina	Tesina Marcello Zin
60	Le Applicazioni nel cloud	Articolo	Le applicazioni nel Cloud
61	Misure di QoS nei sistemi Cloud	Tesi	Misure di QoS nei sistemi Cloud
62	lock-in	definizione	lock-in
63	Multi-tenant	definizione	Multi-tenant
64	SOA	definizione	SOA
65	SLA	definizione	SLA
66	Sviluppo di un sistema di cloud	Tesi	Sviluppo Cloud
67	Raccomandazioni Cloud e PA	Documento	Raccomandazioni nella PA
68	Sviluppo di un Sistema di Cloud Computing	Tesi	Sviluppo Cloud Davide Michelino

Bibliografia

[1] Luis M. Vaquero, Luis Rodero-Merino,Juan Caceres, Maik Lindner, *A Break in the Clouds: Towards a Cloud Definition*, Link, ACM SIGCOMM Computer Communication Review, Volume 39, Number 1, January 2009

[2] Aa.vv, Gartner Newsroom (December 2012), *Gartner Says Cloud Computing Will Be As Influential As E-business*, Link.

[3] Aa.vv,HSPI ,Marzo 2012, *CLOUD COMPUTING REFERENCE ARCHITECTURE*, Link.

[4] Aa.vv, Gartner Newsroom (December 2012), *Predicts 2013: Cloud Computing Becomes an Integral Part of IT*, Link.

[5] B. Sodhi and T. Prabhakar. Cloud-oriented platforms: bearing on application architecture and design patterns. In Proceedings of the 2012 IEEE Eighth World Congress on Services, 2012.

[6] Jonson B., Bobbie Johnson, technology correspondent (Settembre 2008), *Cloud computing is a trap, warns GNU founder Richard Stallman*, Link.

[7] M. Dalheimer and F.-J. Pfreundt. Genlm: *License management for grid and cloud computing environments*. In Cluster Computing and the Grid, Link, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on, pages 132 –139, may 2009.

[8] Spina V., Blade Server (Settembre 2008), *Le "lame" che hanno rivoluzionato il modo di intendere il server rack*, Link.

[9] Amazon. Amazon elastic cloud computing. Link, 2012.

[10] Parlamento Europeo,Studio Cloud Computing (Maggio 2012), Link.

[11] NIST, Mell, P. & Grance, T., *The NIST Definition of Cloud Computing, 2011.*, Link.

[12] Unione Europea, la Commissione europea avanza decisa (Ottobre 2013), Link.

[13] Unione Europea, Cloud Computing Certification Schemes List - CCSL (Febbraio 2014), Link.

[14] OpenStack. OpenStack Compute Developer Guide API v2 Reference, Link

[15] Libvirt3, The virtualization API, Link

[16] N. Borenstein and J. Blake. *Cloud computing standards: Where's the beef?* Internet Computing, IEEE, 15(3):74 –78, May-June 2011.

[17] IBM. Smart cloud enterprise, Link.

[18] C. Ward, N. Aravamudan, K. Bhattacharya, K. Cheng, R. Filepp, R. Kearney, B. Peterson, L. Shwartz, and C.C. Young. *Workload migration into clouds.* Cloud Computing, IEEE International Conference on, 0:164–171, 2010.

[19] Mohammad Hajjat, Xin Sun, Yu-Wei Eric Sung, David Maltz, Sanjay Rao,Kunwadee Sripanidkulchai, and Mohit Tawarmalani, *Cloudward bound: planning for beneficial migration of enterprise applications to the cloud*, Link, SIGCOMM Comput. Commun. Rev., 41:243–254, August 2010.

[20] J. Varia. Architecting for the cloud: best practices, January 2010.

[21] IBM Research, Xiaolan,Amitkumar M.,Akshat, Link, IT migration & modernization.

[22] Achieve Cloud Economics For Operations And Services, Link, John R. Rymer, May 2, 2012

[23] Agenzia per L'italia Digitale, Raccomandazioni e proposte sullitilizzo del cloud computing nella pubblica amministrazione, Link, 28 giugno 2012

[24] C. Fehling, F. Leymann, R. Mietzner, and W. Schupeck. A collection of patterns for cloud types, cloud service models, and cloud-based application architectures, May 2011.

[25] Agenzia per L'italia Digitale, Cloud computing, Link, 03 Febbraio 2015

[26] Consip: al via la gara per servizi Cloud nella Pubblica Amministrazione, Link, Roma 24 dicembre 2013

[27] IBM Global Technology Services, Virtualizzare il disaster recovery utilizzando il cloud computing, Link, Gennaio 2012

[28] C. Fehling, T. Ewald, F. Leymann, M. Pauly, J. Rutschlin, and D. Schumm. Capturing cloud computing knowledge and experience in patterns. In Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing, 2012.

[29] VMWARE,Keith Adams, Ole Agesen,A Comparison of Software and Hardware Techniques for x86 Virtualization, Link, October 21–25, 2006, San Jose, California.

[30] B. Sodhi and T. Prabhakar, *Cloud-oriented platforms: bearing on application architecture and design patterns. In Proceedings of the 2012 IEEE Eighth World Congress on Services*Link, 2012.

[31] S. Bykov, A. Geller, G. Kliot, J. R. Larus, R. Pandya, and J. Thelin. Orleans: cloud computing for everyone., Link, In ACM Symposium on Cloud Computing, October 2011.

[32] F. Chang, J. Dean, S. Ghemawat, W. Hsieh, D. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. Gruber. Bigtable: a distributed storage system for structured data, November 2006.

[33] A. Edmonds, T. Metsch, A. Papaspyrou, and A. Richardson. Toward an open cloud standard. Internet Computing, IEEE, August 2012.

[34] R. Karmani and G. Agha. Actors, Link, 2011.

[35] W. Kim. Cloud architecture: A preliminary look. In Proceedings of the 13th International Conference on Information Integration and Web- based Applications and Services, Link.

[36] R. Moreno-Vozmediano, R. Montero, and I. Llorente. Iaas cloud architecture: From virtualized data centers to federated cloud infrastructures, February 2011.

[37] Giovanni Lofrumento, Articolo Notiziario tecnico Telecom Italia, *Servizi le applicazioni nel Cloud: Opprotunità e prospettive*, 2011 Link.

[38] Marcello Zin,Tesi, *Cloud Computing: Analisi dei modelli architetturali e delle tecnologie per lo sviluppo di applicazioni*, Bologna 2012 Link.

[39] Marco De Canal,Tesi, *Cloud Computing analisi dei modelli architetturali.*, Bologna 2012 Link.

[40] Dovik Çoba,Tesi,*Misure di QoS nei sistemi Cloud: Stato dell'Arte*,Padova 2013 Link.

[41] Davide Michelino,Tesi di Laurea, *Sviluppo di un Sistema di Cloud Computing Federato con supporto per la contestualizzazione*,Napoli 2013 Link.