UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI INGEGNERIA

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica

STUDIO DELLE TECNOLOGIE PER IL CLOUD COMPUTING INFRASTRUCTURE-AS-A-SERVICE E REALIZZAZIONE DI UN SERVIZIO DI AUTOSCALING SESSION-AWARE

ANNO ACCADEMICO 2010-2011

Relatore:
Prof. Massimo Maresca
Correlatore:
Ing. Michele Stecca

Laureando:
Luca Bazzucco
Mat: 604133-IF

Alla mia Famiglia ...

Ringraziamenti

Scrivere una paginetta di ringraziamenti non è così semplice come sembra, un po' perché le parole tendono a mancare, e anche perché si ha sempre il terrore di dimenticare qualcuno di importante. Mi pare doveroso però ringraziare tutte le persone che mi sono state vicine in questi cinque anni universitari.

Innanzitutto ringrazio gli amici del liceo, Emanuele, Francesco, Giacomo e Nicolò, che si sono dovuti ascoltare tutte le mie avventure, e disavventure, universitarie ogni finesettimana.

Un pensiero particolare va a tutti gli amici e colleghi ingegneri: Davide, Federica, Federico, Lucio, Mauro, Massimiliano e Michele, grandi compagni di studio e di pause caffè. Senza di loro non sarei riuscito a raggiungere i risultati che ho ottenuto, ma soprattutto questi cinque anni non sarebbero stati così divertenti.

Desidero ringraziare il mio relatore Prof. Massimo Maresca, per la sua disponibilità e per avermi fornito preziosi consigli, non solo per lo sviluppo di questa tesi.

Un ringraziamento speciale va ai ricercatori del laboratorio Sintesi: Martino Fornasa e in particolare Michele Stecca, per avermi seguito ogni giorno durante questi ultimi mesi, guidandomi nella redazione della tesi e aiutandomi a risolvere tutti i problemi che si sono presentati.

Un grazie ai parenti di Nogara, in particolare a mia nonna Giuliana, che mi aiuta ogni giorno a "disintossicarmi" dalla mensa universitaria, e a mio zio Stefano, che per primo mi ha trasmesso la passione per i computer.

Un ringraziamento va anche agli zii, ai cugini e ai nonni di Correzzo, che hanno sempre avuto un pensiero per me e mi hanno sostenuto durante il mio percorso di studi.

A mio fratello Alberto, che ha anche lui da poco intrapreso la sua carriera universitaria, va un sincero "in bocca al lupo".

Il ringraziamento più grande però va ai miei genitori, Marta e Renato, che mi hanno dato la possibilità di studiare con serenità e hanno sempre appoggiato le mie scelte, consigliandomi ogni volta la strada giusta da prendere.

Ringrazio infine tutte le persone che in qualche modo mi sono state vicine in questi cinque anni e hanno creduto in me.

Grazie di cuore!

INDICE

1. INTRODUZIONE

L'evoluzione tecnologica e l'aumento dell'utilizzo di sistemi informatici nelle attività quotidiane e aziendali stanno trasformando le risorse dell'*Information and Communication Technology (ICT)* in servizi di massa: esse, infatti, vengono ormai commercializzate e fornite in modo analogo alle risorse "tradizionali" quali acqua, luce, gas e telefonia, secondo il paradigma del pagamento a consumo [1]. Questo fenomeno, inevitabilmente, porta con sé delle conseguenze; per rendere i servizi informatici una risorsa di massa a tutti gli effetti, è necessario che i fornitori offrano due garanzie agli utenti: la possibilità di utilizzare la quantità di risorse richiesta in ogni momento e il mantenimento ininterrotto di elevati livelli di disponibilità e accessibilità del servizio. Questo nuovo modello di utilizzo delle risorse informatiche prende anche il nome di *Utility Computing*, termine coniato già negli anni '70 da Leonard Kleinrock [2], uno dei principali contribuenti del progetto *Advanced Research Projects Agency Network (ARPANET)*, che pose le basi per lo sviluppo della rete Internet di oggi: ``*As of now, computer networks are still in their infancy, but as they grow up and become sophisticated, we will probably see the spread of `computer utilities' which, like present electric and telephone utilities, will service individual homes and offices across the country''.*

Oggi, non tutti i servizi che forniscono risorse informatiche adottano il paradigma dell'*utility computing:* di solito si segue un approccio più tradizionale, utilizzato fin dai primi sviluppi delle tecnologie informatiche. In tale paradigma, le macchine fisiche vengono collocate in uno o più data center gestiti direttamente dal personale aziendale, che si occupa del mantenimento sia dell'hardware sia del software installato. Una soluzione di questo tipo è stata l'unica possibile fino a qualche anno fa: a causa, soprattutto, della mancanza di collegamenti di rete veloci e affidabili, era necessario infatti che le macchine fossero collocate fisicamente vicino agli utilizzatori, in modo da garantire prestazioni adeguate alle applicazioni in esecuzione. Negli ultimi anni invece, con lo sviluppo delle reti e la sempre maggiore disponibilità di banda, è diventato possibile utilizzare da remoto apposite macchine messe a disposizione da servizi di *hosting* e *housing*, i quali si occupano del collocamento e della gestione fisica delle stesse. Oggi molte aziende utilizzano servizi di questo tipo, evitando così di dover sostenere i costi e occuparsi del collocamento, dell'alimentazione e della climatizzazione del data center. I sistemi di

pagamento si basano su abbonamenti di durata mensile o annuale stipulati con i clienti, a seconda della tipologia di servizio offerta. Le soluzioni di *hosting* e *housing* risultano oggi le più utilizzate, in particolare dalle piccole e dalle medie aziende; tuttavia esse presentano una limitazione: costringono gli utenti a dimensionare in maniera preventiva le proprie risorse IT, cercando di prevedere quanto grande sarà il carico di lavoro da svolgere e come si dovranno gestire eventuali picchi di richieste. I data center infatti sono spesso sovradimensionati rispetto alla quantità di lavoro medio che devono svolgere, consentendo così al sistema di poter soddisfare un eventuale aumento di richieste, senza creare disservizi ai propri clienti. Tali fenomeni di *overprovisioning* sono sicuramente sfavorevoli dal punto di vista economico, dato che comportano spese fisse, effettivamente necessarie solamente quando si verifica un aumento del lavoro da svolgere, mentre durante i periodi in cui l'utilizzazione di risorse è minore, alcune di esse resteranno inutilizzate. D'altra parte, non è conveniente dimensionare le risorse in base al carico medio: il sistema infatti non funzionerebbe correttamente qualora si presentasse un picco di richieste, presentando una situazione di *underprovisioning* causando, di conseguenza, disservizi per gli utilizzatori del sistema [3].

Per ovviare a questo problema, sono state sviluppate le nuove tecnologie di *Cloud Computing*. I servizi offerti si basano sul concetto di "utility computing": le risorse IT vengono fornite in base alle specifiche richieste dei clienti, sfruttando la rete Internet per la distribuzione e applicando tariffe a consumo [4]. Tuttavia, la peculiarità dei servizi di *Cloud Computing* sta nella loro scalabilità ed elasticità: ogni utente può ottenere la quantità di risorse di cui necessita, che vengono rese immediatamente disponibili e pronte per essere utilizzate, non appena vengono richieste. In particolare, non è più necessario l'*overprovisioning*: si possono richiedere esclusivamente le risorse necessarie per eseguire le proprie operazioni, poiché in ogni momento è possibile aumentarne o diminuirne la quantità allocata, qualora la situazione lo richiedesse. Il sistema di pagamento a consumo permette agli utenti di calcolare le proprie spese solamente in base alla quantità di risorse effettivamente utilizzata, evitando così tutte le problematiche dei servizi basati su contratti ad abbonamento.

Le risorse offerte dai provider Cloud sono molteplici e si possono dividere in tre categorie: *Software-as-a-Service*, in cui vengono offerte una o più applicazioni pronte all'uso, alle quali gli utenti accedono tramite un browser web; *Platform-as-a-Service*, in cui viene messa a disposizione una piattaforma elastica e scalabile sulla quale ogni utente può sviluppare le proprie applicazioni; *Infrastructure-as-a-Service* (*IaaS*), in cui vengono fornite risorse fisiche, completamente gestibili e personalizzabili dall'utente, come capacità di calcolo o spazio di storage [5]. Tutti questi servizi devono assicurare alti livelli di disponibilità, dato che le risorse offerte dai *cloud provider* potrebbero essere di fondamentale importanza per il core business di un cliente; è quindi essenziale che i *cloud provider* forniscano delle garanzie sufficienti sulle prestazioni e sulla disponibilità del servizio nel corso dell'anno. Tali garanzie vengono rilasciate mediante dei *Service Level Agreement* (SLA) stipulati tra il fornitore del servizio e il cliente. Oggi la disponibilità di servizio offerta dai *cloud provider* permette alla maggior parte delle aziende di poter utilizzare per il proprio business soluzioni di *Cloud Computing*, dato che viene

solitamente garantito il 99.9% di *uptime* del servizio durante l'anno solare: una percentuale sufficientemente elevata per la maggior parte delle applicazioni.

L'analogia fatta con la "nuvola" indica che gli utenti possono collegarsi tramite Internet e richiedere i servizi offerti dal *cloud provider*, ma non possono conoscere e interagire con l'infrastruttura (hardware e software) sottostante, utilizzata per realizzare effettivamente il servizio: essa infatti viene "nascosta" dalla nuvola posta sopra di essa. Non conoscere come viene implementato un servizio non è da considerarsi una limitazione: permette infatti di semplificare le operazioni dell'utente, che si può concentrare sul proprio core business, senza doversi preoccupare della gestione dell'infrastruttura utilizzata per l'implementazione del servizio. Uno dei vantaggi più pubblicizzati da tutti i *cloud provider* è proprio la semplicità di utilizzo, garantita appunto dalla presenza della "nuvola" che nasconde le attività di implementazione e gestione del servizio. Oltre a nascondere l'infrastruttura, i *cloud provider* devono essere in grado di mantenere i propri servizi ridondanti e affidabili in caso di guasti: per questo, la maggior parte delle risorse hardware utilizzate dai più grandi fornitori di servizi cloud, come Amazon [6], Microsoft [7], Google [8] e IBM [9], vengono dislocate in data center collocati in luoghi diversi, assicurando un alto livello di ridondanza dei dati e di tolleranza ai guasti.

La tecnologia utilizzata dai provider cloud per la realizzazione dei propri servizi si basa su nuove funzionalità recentemente introdotte nei processori, che consentono di eseguire applicazioni su macchine virtuali (VM) mantenendo degli ottimi livelli di prestazioni finali [10]. Implementando un servizio mediante la tecnologia della virtualizzazione si possono soddisfare nello stesso momento due dei requisiti fondamentali richiesti per fornire un servizio cloud: l'*isolamento delle risorse* offerte dall'hardware utilizzato e *l'isolamento tra le diverse macchine virtuali* in esecuzione. Quest'ultima caratteristica permette di ottenere una separazione delle risorse allocate tra i vari utenti, garantendo elevati livelli di sicurezza e di qualità del servizio, impedendo che le operazioni eseguite da un utilizzatore possano interferire con quelle eseguite da altri. Le tecnologie di virtualizzazione, oltre a garantire il livello di isolamento appena descritto, consentono ai *cloud provider* di allocare le risorse disponibili in modo da massimizzarne l'utilizzo: è possibile, ad esempio, assegnare una risorsa fisica a più utenti, gestendone poi gli accessi concorrenti in modo da evitare interferenze. Grazie a questi meccanismi, per un *cloud provider* si presenta la possibilità di risparmiare considerevolmente sulle spese dell'hardware, migliorando di conseguenza l'offerta con tariffe più basse e competitive.

I *cloud provider* devono utilizzare un sistema di gestione delle risorse *market oriented*, in modo da poter soddisfare richieste variabili sia dal punto di vista della quantità di risorse, che della qualità del servizio [11]. Tutti questi accorgimenti non sono solamente utili al *cloud provider* per ridurre le spese e per massimizzare il profitto, ma possono indirettamente favorire anche gli stessi clienti, che potranno beneficiare di eventuali abbassamenti di prezzo, visti i costi limitati del fornitore, o di miglioramenti nella qualità e nelle prestazioni del servizio offerto.

Utilizzare i servizi di *Cloud Computing* comporta una serie di vantaggi all'utente finale, in particolare grazie al fatto che le risorse possono essere richieste e ottenute *on-demand*, nella quantità desiderata. In questo modo si consente agli utenti di eliminare tutte le spese relative all'acquisto e alla gestione dell'hardware e del software, convertendole in spese operative (Capex), che vengono calcolate in base al reale utilizzo delle risorse. A questi vantaggi economici si aggiungono la flessibilità e l'elasticità dei servizi offerti, in grado di adeguarsi alle esigenze di tutte le categorie di utenza, a partire dalle startup fino ad arrivare alle grandi aziende [12].

Vanno tenuti presente, però, anche alcuni fattori che possono limitare le possibilità di utilizzo dei servizi Cloud: in primo luogo si deve considerare la dipendenza da terzi per la fornitura di risorse spesso cruciali per il proprio business; inoltre, è da valutare anche come gli eventuali clienti possano reagire alla decentralizzazione dei servizi da loro utilizzati e dei dati ad essi collegati [13]. A queste considerazioni si devono aggiungere anche le limitazioni che vi sono dal punto di vista legale: in alcuni Stati si richiede che tutti gli archivi contenenti dati sensibili siano protetti mediante sistemi che soddisfino determinati requisiti o addirittura che i server contenenti tali informazioni siano collocati fisicamente all'interno dei confini dello Stato o dell'Unione Europea. Per quanto riguarda la disponibilità e la qualità del servizio, i *cloud provider* sono in grado di assicurare ottimi livelli di prestazioni e percentuali di *uptime* elevate, tuttavia alcune applicazioni presentano requisiti che i *cloud provider* oggi non sono ancora in grado di soddisfare. Un esempio è costituito da tutte quelle applicazioni in ambito governativo o finanziario, che non possono permettersi di avere periodi di downtime o di non disponibilità del servizio. Spesso, in queste situazioni, la soluzione migliore è ancora quella di implementare le applicazioni *in-house*, in modo da poter gestire tutte le operazioni direttamente, senza dover far affidamento a società di terze parti. Bisogna sottolineare però che nella maggior parte dei casi i fornitori di servizi cloud sono in grado di offrire garanzie di servizio uguali, se non maggiori, a quelle di sistemi gestiti autonomamente o di servizi di *housing* e *hosting*, consentendo così agli utenti di scegliere di utilizzare le risorse di tipo *IaaS* per eseguire le proprie applicazioni.

Struttura della Tesi

Nella prima parte di questa tesi sono stati analizzati nel dettaglio tutti gli aspetti necessari per valutare quali soluzioni convenga utilizzare in ogni situazione e se sia conveniente sfruttare i servizi offerti da un *cloud provider*. L'attenzione è stata posta principalmente nel settore del C*loud Computing Infrastructure-as-a-Service*, in cui le risorse fornite sono costituite da macchine virtuali, completamente gestibili e personalizzabili dall'utente. Nel capitolo 2 viene data una definizione di *Cloud Computing*, descrivendo le diverse tipologie di servizi offerti; nel capitolo 3 viene presentata un'analisi più approfondita delle offerte di tipo *Infrastructure-as-a-Service;* nel capitolo 4, si analizza la specifica offerta del *cloud provider* Amazon; nel capitolo 5 viene presentata una soluzione open source, *Eucalyptus* [14], che permette di realizzare in servizio di tipo IaaS utilizzando le proprie risorse fisiche; nel capitolo 6 vengono presentati software, commerciali e opensource, che implementano lo stesso tipo di servizio.

Nella seconda parte della tesi è stato analizzato uno dei servizi più interessanti forniti dal *cloud provider* Amazon, ossia *Autoscaling* [15], di cui viene data una descrizione dettagliata nel capitolo 7. Il servizio permette di definire un *cluster* di macchine virtuali, il cui numero si adatta in maniera automatica al carico di lavoro, controllando periodicamente l'utilizzo di risorse quali la percentuale di CPU o la quantità di dati trasmessi dalle macchine in esecuzione. Sfruttando questa funzionalità uno sviluppatore può creare un'applicazione in grado di scalare fino a raggiungere grandi dimensioni, in base alla quantità di risorse richieste. Nello stesso tempo è possibile evitare di sovradimensionare il numero di risorse allocate, assicurandosi la possibilità di soddisfare tutte le richieste dei propri clienti, dato che il provider IaaS consente di aggiungere e rimuovere macchine virtuali in tempi molto brevi. Un servizio come Autoscaling si adatta perfettamente a tutte quelle applicazioni che hanno carichi di lavoro molto variabili o imprevedibili, sia nell'arco di una giornata che durante l'anno: molto spesso infatti si richiede di poter soddisfare le richieste che provengono da un gran numero di utenti in determinati intervalli di tempo, come può succedere per i portali di e-commerce durante il periodo natalizio o per un sito di un giornale quando si verifica un evento importante. È necessario quindi che il sistema sia in grado di adattarsi a tali aumenti di richieste in modo da evitare disservizi per i propri clienti.

Il servizio offerto da Amazon è molto funzionale, tuttavia non è in grado di gestire alcune operazioni che potrebbero essere richieste da sviluppatori di applicazioni, in particolare di web application: non è possibile, infatti, garantire un supporto affidabile alla persistenza delle sessioni a livello applicativo. Quando il sistema effettua l'operazione di terminazione di un'istanza, a causa di un calo del carico di lavoro registrato, non verifica se vi sono delle sessioni attive associate alla macchina virtuale da spegnere. Il sistema rischia di perdere i dati associati ad alcuni utenti, i quali vedranno terminare improvvisamente le loro sessioni. Nel capitolo 9 viene descritto come è stato progettato un servizio analogo ad Autoscaling per il software Eucalyptus, realizzando non solo le funzionalità analoghe alla soluzione proposta da Amazon, ma garantendo anche il supporto completo alla persistenza delle sessioni. Il sistema è in grado di interagire con *Eucalyptus* e con un load balancer software per poter monitorare lo stato delle sessioni attive e decidere di conseguenza come eseguire le operazioni richieste per le procedure di "scaling", come l'aumento e la diminuzione del numero di macchine virtuali attive del cluster. L'implementazione di questo servizio viene descritta nel capitolo 10, in cui vengono presentate alcune parti del codice sorgente del software sviluppato.

2. CLOUD COMPUTING

Dare una definizione precisa di Cloud Computing non è facile, in rete infatti se ne possono trovare molteplici, anche se tutte leggermente diverse tra di loro. *Gartner Group*, uno dei maggiori analisti dell'Information Technology, definisce il Cloud Computing in questo modo: *"a style of computing in which scalable and elastic IT enabled capabilities are delivered as a service to external customers using Internet technologies"*. Nonostante non si sia ancora raggiunto un accordo in letteratura su quali debbano essere le caratteristiche necessarie per definire un servizio di Cloud Computing, si può fare riferimento ai cinque attributi, definiti sempre da Gartner, nell'articolo "Gartner Highlights Five attributes of Cloud Computing" [4]:

- *Essere basato sui servizi*: i servizi offerti vengono acceduti dagli utenti tramite interfacce ben definite, spesso effettuando richieste a servizi web. Tali interfacce nascondono i dettagli implementativi e rendono possibile al fornitore del servizio di poter rispondere in maniera diretta all'utente. Ogni servizio viene infatti considerato "ready-to-use", visto che è stato progettato e sviluppato per soddisfare le specifiche necessità degli utenti, non è quindi importante quale sia il background tecnologico utilizzato per implementare tale servizio, ma l'attenzione è focalizzata sulle funzionalità che un utente può utilizzare. Le caratteristiche fondamentali che si devono garantire sono: alta disponibilità, tempi di risposta brevi e un ottimo rapporto prezzo-prestazioni. L'attenzione quindi è posta sulle funzionalità offerte dal servizio, non su come questo sia effettivamente realizzato dal punto di vista tecnologico.
- *Essere scalabile ed Elastico:* i servizi cloud devono essere in grado di scalare sia vero l'alto che verso il basso, in base alle richieste dell'utente, in maniera completamente automatica; i tempi di risposta possono variare da pochi secondi a qualche ora, a seconda del tipo di servizio offerto. Per quanto riguarda la Scalabilità, si intende che l'infrastruttura sottostante è in grado di gestire un aumento delle richieste da parte dell'utente in maniera efficiente e veloce, mentre per Elasticità si intende non solo la capacità di aumentare e ridurre le risorse fornite all'utente, ma anche una flessibilità dal punto di vista del modello economico utilizzato. Riassumendo, è necessario che il servizio possa scalare in base alle richieste aggiungendo o rimuovendo risorse

in base allo stato del sistema, lasciando completa libertà all'utente nella scelta di quante e quali risorse utilizzare.

- *Condiviso*: i servizi condividono un pool di risorse per costruire un'economia di scala, in modo da poter utilizzare le risorse IT con la massima efficienza. L'infrastruttura sottostante, il software o le piattaforme utilizzate sono condivise dagli utilizzatori del servizio. Questa scelta implementativa consente di sfruttare tutte le risorse disponibili per soddisfare le richieste dei vari utenti, cercando di mantenere le tariffe ridotte.
- *Utilizzare modelli di pagamento a consumo*: i servizi offerti vengono fatturati ai clienti utilizzando delle metriche che consentono di utilizzare diversi modelli di pagamento. Il service provider deve avere un sistema per monitorare l'utilizzo di risorse di ogni servizio offerto, in modo da poter rendere disponibile metodi di fatturazione diversi tra loro: le tariffazioni più utilizzate seguono il modello pay-as-you-go, anche se sono presenti sistemi di pagamento su base mensile o addirittura gratuiti in determinate condizioni. Tutti questi metodi di pagamento si devono basare sull'effettiva quantità di risorse utilizzate dall'utente, che possono essere calcolate in base al numero di ore per cui una risorsa è stata occupata, alla quantità di dati trasferiti o ad altre metriche di utilizzo.
- *Utilizzare le tecnologie Internet*: i servizi cloud vengono distribuiti utilizzando i protocolli e le richieste strutturate già sviluppate per le altre tecnologie Internet. Per accedere ai servizi cloud si usa solitamente il protocollo HTTP inviando le richieste agli specifici url associati al servizio cloud offerto. La maggior parte dei service provider espone dei servizi web accessibili seguendo le specifiche WSDL.

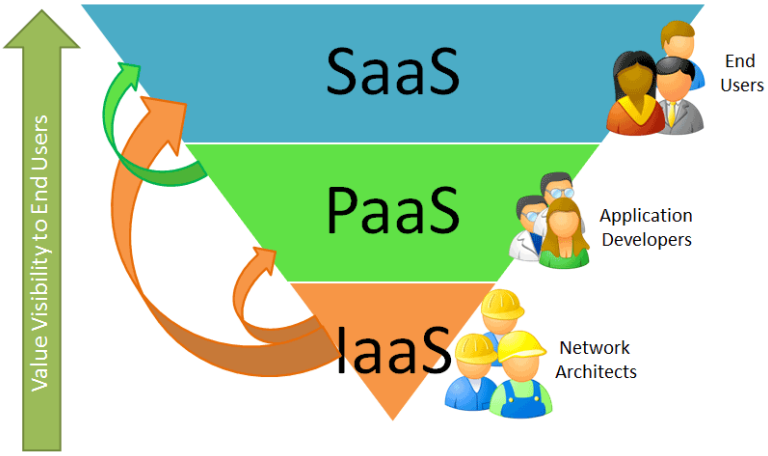


Figura 1 Le tre sottocategorie del cloud computing

Le risorse IT che possono essere offerte dai *cloud provider* si possono dividere in tre categorie, divise in base alla tipologia di servizio offerto, in particolare in base al livello di astrazione che si ottiene. Le offerte cloud si possono dividere in *Software-as-a-Service*, che è in grado di fornire le funzionalità di un applicativo vero e proprio, *Platform-as-a-Service*, che fornisce una piattaforma per lo sviluppo delle

applicazioni sviluppate dall'utente e *Infrastructure-as-a-Service*, che fornisce risorse hardware utilizzabili dall'utente in base alle proprie necessità [16].

2.1 Software as a Service - SaaS

Le offerte di tipo *Software-as-a-Service* sono oggi le più popolari e utilizzate soluzioni di cloud computing, in particolare i servizi più utilizzati sono i client di posta elettronica come Yahoo mail, e Gmail, ma sono presenti anche programmi come office suite, quali text-editor e spreadsheet, o software di Customer Relationship Management.

Le caratteristiche che accomunano tutti i tipi di servizi di Software-as-a-Service sono per prima cosa la modalità di accesso: infatti viene utilizzato un browser web per facilitare le comunicazioni client server e per gestire al meglio la presentazione dei contenuti. Le elaborazioni però vengono eseguite nel data center in cui risiedono i server dedicati a eseguire il software in questione. Ogni utente può quindi accedere alla propria applicazione indipendentemente dalla macchina dalla quale sta lavorando, fornendo le proprie credenziali, anche i dati vengono memorizzati sui server remoti, rendendoli accessibili in ogni momento. I servizi di tipo SaaS sono spesso forniti gratuitamente, come applicazioni di web mail o le office suite online, sono presenti anche servizi più avanzati, destinati principalmente a utenti di tipo business, che richiedono pagamenti mensili, con eventuali sovrapprezzi se l'utente richiede funzionalità aggiuntive. L'utilizzo di applicazioni di tipo SaaS comporta una serie di *vantaggi* per l'utente finale:

- L'utilizzo dell'applicazione non è più vincolato alla macchina su cui essa è installata, ma è accessibile in ogni momento tramite Internet
- Non si richiedono operazioni di manutenzione del software, infatti tutte queste attività dovranno essere eseguite dall'amministratore del servizio e saranno completamente trasparenti all'utente. In questo modo è garantito che si utilizzerà sempre la versione più aggiornata del software
- I dati sono centralizzati ed accessibili tramite la rete, si evita quindi di dover effettuare copie per spostarli da una macchina all'altra
- Solitamente si può scegliere quale livello di servizio si vuole utilizzare, pagando quindi unicamente per i servizi di cui si necessita

A questi aspetti positivi però si devono associare alcune problematiche che emergono nell'utilizzo di piattaforme di tipologia SaaS, tali problematiche sono legate essenzialmente alla disponibilità e alla sicurezza del servizio:

- Il fatto di non dover gestire la manutenzione del software non permette all'utente di avere il controllo su quale versione del programma utilizzare e non permette evitare eventuali malfunzionamenti che si possono presentare
- Potrebbe succedere che i server su cui viene eseguita l'applicazione non siano più raggiungibili, rendendo quindi inutilizzabile il servizio per l'utente

- I dati vengono memorizzati su server remoti, nel caso questi siano dotati di sistemi di sicurezza inefficienti potrebbe essere possibile per malintenzionati accedere ai dati personali dell'utente
- Un'ultima problematica riguarda la mancanza di garanzia della continuità del servizio, potrebbe infatti accadere che il provider decida di smettere di fornire il software utilizzato, per motivi economici o scelte di mercato

Ad oggi, tutti i fattori positivi sono sicuramente più importanti rispetto alle problematiche che possono emergere, infatti la maggior parte queste si presentano quanto il service provider non ha capacità sufficienti, economiche o tecniche, per poter gestire il servizio. L'unico accorgimento che si deve avere è quello di scegliere un service provider fidato e che possa garantire dei livelli di disponibilità del servizio e di sicurezza sufficientemente elevati; oggi la maggior parte dei service provider di SaaS soddisfano questi requisiti e consentono agli utenti di poter usufruire dei loro servizi in maniera totalmente sicura.

2.2 Platform as a Service – PaaS

Ad un livello più basso di astrazione si trovano i servizi del tipo *Platform-as-a-Service* o *PaaS*, essi offrono una piattaforma su cui gli utenti possono sviluppare le proprie applicazioni. A partire da questo tipo di servizi Cloud si richiedono agli utenti delle conoscenze tecniche più specifiche, da qui in avanti si dovrà parlare di sviluppatori più che di utenti generici. Utilizzando i servizi PaaS gli sviluppatori non devono preoccuparsi delle configurazioni del sistema a basso livello, come il sistema operativo, i sistemi di storage e di hosting, di cui dovrà occuparsi il service provider. Vengono forniti degli strumenti di amministrazione molto semplici, che consentono di effettuare l'upload del codice sorgente e di gestire l'esecuzione della applicazione.

Il servizio consente quindi agli utenti di sviluppare le proprie applicazioni e poi *deployarle* (eseguirle e renderle pubbliche sulla rete internet) utilizzando la piattaforma fornita dal service provider, spesso i servizi di PaaS sono compatibili con specifici linguaggi di programmazione e IDE, che gli sviluppatori sono obbligati ad utilizzare.

Sarà compito del service provider configurare appositamente l'hardware e il software necessario per l'esecuzione delle applicazioni; inoltre dovrà mantenere aggiornato il software della piattaforma e garantire la disponibilità di banda richiesta dagli utenti. Uno dei vantaggi dell'utilizzo di servizi PaaS è appunto quello di poter di disporre di risorse virtualmente illimitate, infatti il service provider configurerà il sistema in base al carico di lavoro attuale, mentre lo sviluppatore non dovrà né modificare l'applicazione, né effettuare operazioni di manutenzione di alcun tipo. Molti service provider PaaS limitano alcune funzionalità che potrebbero essere utilizzate dal codice delle applicazioni, in modo da evitare di dare accesso al sistema operativo sottostante, non consentendo utilizzi malevoli di tali funzionalità. Gli sviluppatori devono rispettare questa tipologia di vincoli, altrimenti le applicazioni non risulteranno funzionanti o utilizzabili.

Le modalità di pagamento sono determinate dalla quantità di risorse consumate dall'applicazione sviluppata, solitamente vengono considerati lo spazio su disco occupato e il traffico generato, il numero di pagine accedute o la quantità di dati trasmessi.

Esempio di provider PaaS – Google App Engine

Google App Engine [8] è stata una delle prime offerte di tipo PaaS, supporta il linguaggio Python e applicazioni web scritte in linguaggio Java-jsp, sviluppate sul modello utilizzato dall'application server Tomcat di Apache [17]. Vengono forniti anche degli strumenti per sviluppare le applicazioni in locale, una volta completate le fasi di testing è sufficiente premere un pulsante per trasferire l'intera applicazione sulla piattaforma di App Engine. Una volta messa in esecuzione l'applicazione essa potrà essere acceduta via web, utilizzando un dominio comune (appspot.com) oppure uno personalizzato, se l'utente lo richiede.

Google App Engine è gratuito se si utilizzano fino ad un massimo di 1GB di spazio su disco, 1GB di dati trasferiti e 43 milioni di richieste al mese, se si superano questi valori si dovrà pagare in base al consumo aggiuntivo effettuato. Sono presenti diversi metodi di fatturazione basati, oltre ai parametri appena descritti, anche sul numero di chiamate a determinate API e alla quantità di tempo per cui si è occupata la CPU; un utente può anche definire dei limiti massimi giornalieri di spesa, al fine di poter porre un tetto massimo alle proprie spese. Tutti i dettagli sulla tariffazione e sulle funzionalità di Google App Engine si possono consultare all'indirizzo: http://code.google.com/intl/it-IT/App Engine/docs/

Altri provider di servizi PaaS sono *Microsoft Azure* (ASP), *Force.com*, *Engine Yard* (Ruby on Rails), *Aptana Cloud* (PHP, Ruby e Javascript) ed altri, come si nota ognuno di essi supporta un ben determinato tipo di ambiente di sviluppo, dovranno essere quindi gli utenti a scegliere quale provider possa essere compatibile con le proprie esigenze.

2.3 Infrastructure as a Service – IaaS

I servizi cloud di tipo *Infrastructure-as-a-Service* si collocano al livello più basso di astrazione tra le diverse tipologie di servizi cloud, essi forniscono una di serie di risorse hardware in base alla richieste dell'utente, che ne avrà un controllo completo e potrà gestire in maniera autonoma come utilizzare le risorse fornite dal *cloud provider*. Le tipologie di risorse che vengono offerte sono sostanzialmente tutte quelle che sono necessarie per l'esecuzione di una macchina fisica, ossia CPU, memoria, spazio su disco e connettività alla rete; vengono anche offerti servizi complementari, come la gestione delle policy di sicurezza e il monitoraggio delle risorse utilizzate. La maggior parte dei servizi di tipo IaaS fornisce delle macchine virtuali, le cui caratteristiche coincidono con la quantità di risorse richieste dall'utente; le modalità con cui viene eseguita effettivamente ogni macchina virtuale vengono gestite dal service provider e non sono visibili all'utente finale, a cui vengono forniti solamente gli strumenti per poter accedere a tale macchina, da qui in poi denominata anche *istanza*.

Ogni service provider IaaS espone delle interfacce, solitamente utilizzabili tramite linea di comando o web-ui, che consentono ad ogni utente di richiedere l'avvio delle istanze di cui ha bisogno; il sistema si occuperà in maniera autonoma, e non visibile all'utente, di allocare le risorse nel data center dedicato e di effettuare le operazioni per rendere la macchina virtuale accessibile tramite Internet. Si possono scegliere, oltre alla quantità di risorse allocate per ogni istanza, anche quale sistema operativo utilizzare; solitamente i server provider rendono disponibili delle immagini di sistemi operativi già pronte all'uso, con installate le principali distribuzioni linux o windows, ogni utente però può scegliere di utilizzare una propria versione personalizzata del sistema operativo oppure modificarne una di quelle predefinite. Utilizzando un servizio IaaS si può ottenere una macchina virtuale completamente personalizzabile in tutti i suoi parametri, sia hardware che software, di conseguenza questo tipo di soluzione si presta molto bene per tutte quelle situazioni che richiedono una gestione completa delle applicazioni che si vogliono eseguire. Ovviamente il livello così basso di astrazione richiede che l'utente abbia le conoscenze necessarie a gestire l'installazione e l'esecuzione del software, di conseguenza la tipologia di utilizzatori di servizi IaaS è solitamente costituita da sviluppatori di applicazioni, che hanno conoscenze avanzate nel settore. Anche in questo caso il sistema di pagamento si basa sul modello *pay-as-you-go*, in cui vengono calcolate le ore-macchina che ogni utente consuma, il cui prezzo varia a seconda della quantità di risorse allocate per ogni macchina virtuale, a cui si aggiungono i costi relativi alla quantità di spazio occupato, calcolato in GB/mese e alla quantità di traffico generata in ingresso e in uscita dall'istanza.

Nelle sezioni seguenti verranno analizzate nel dettaglio le caratteristiche e i vantaggi offerti dalle soluzioni IaaS, oltre a presentare una serie di scenari di utilizzo e di casi di studio reali, visto che l'attenzione dei capitoli successivi è posta proprio sul tipo di servizio Cloud appena introdotto.

2.3.1 Vantaggi dell'Infrastructure-as-a-Service

Un utente di servizi di IaaS è nella maggior parte dei casi un'azienda, visto che le soluzioni proposte si adattano molto bene ad essere utilizzate per fornire servizi IT, che possono essere utilizzati internamente o esposti al pubblico su Internet. I vantaggi di poter utilizzare una soluzione orientata ai servizi sono principalmente legati al fatto di non dover gestire e mantenere le risorse internamente, ma delegare questi compiti al service provider.

Riduzione dei costi di gestione

In un'azienda la gestione e il mantenimento dei sistemi informatici richiede molte risorse, sia umane che monetarie, si richiede infatti in un primo momento una serie di *investimenti iniziali* molto importanti, per l'acquisto e il collocamento di tutte la macchine fisiche in un data center dedicato; oltre a questo tipo di investimenti la gestione interna delle risorse IT richiede anche di sostenere delle spese per *l'energia di alimentazione e climatizzazione*, oltre ai costi di eventuali interventi di manutenzione in caso di malfunzionamenti dell'hardware [18]. Un'ultima categoria di spese che un'azienda deve affrontare sono quelle necessarie per la *gestione del software* dell'intero data center, bisogna infatti

assumere personale specializzato che si occupi di mantenere operative e funzionanti le applicazioni installate.

Se si utilizza un servizio di Infrastructure-as-a-Service, tutte le spese appena elencate non devono più essere sostenute, si richiede solamente di corrispondere al service provider il costo relativo alle risorse effettivamente utilizzate. Si eliminano quindi tutti costi operazionali dell'hardware, sostituendoli con le spese orarie di utilizzo delle istanze, si vedrà successivamente che a parità di utilizzo di risorse i costi sono circa gli stessi rispetto a una soluzione in-house, ma con una soluzione di tipo IaaS ogni azienda ha la possibilità di dimensionare il numero di risorse utilizzate in ogni momento, riuscendo così a deallocare le risorse non utilizzate in alcuni periodi, riallocandole quando invece la situazione lo richiede [19].

Elasticità, evitare fenomeni di Underprovisioning e Overprovisioning

Una delle caratteristiche peculiari dei servizi IaaS, come del resto di tutti i servizi di tipo Cloud, è quella di essere *elastici*, che nel caso specifico dei servizi infrastrutturali si traduce nella possibilità di allocare on-demand e in tempi molto brevi la quantità di risorse che si desidera. Sfruttando questa funzionalità si possono risolvere alcune problematiche tipiche di alcune applicazioni, soprattutto se queste sono state sviluppate per interagire con gli utenti tramite il Web o, più in generale, tutte quelle applicazioni che non possono prevedere con certezza il carico di lavoro che devono sopportare.

Quando si implementano tipologie di applicazioni soggette a picchi di richieste, come può essere un servizio di e-commerce, se si sceglie una soluzione tradizionale è necessario dimensionare le capacità dei propri server non per riuscire a gestire il livello di carico medio, ma si devono poter soddisfare anche i picchi di richieste che si possono verificare. In alcuni casi si devono valutare anche l'aumento del carico di lavoro che si presentano in certi particolari mesi dell'anno, come ad esempio il periodo natalizio per i servizi di e-commerce. In queste situazioni si devono quindi valutare due situazioni opposte che possono accadere [3]:

- *Overprovisioning*: è stata allocata una quantità troppo elevata di risorse rispetto al carico di lavoro generato dalle applicazioni, di conseguenza tutti i costi relativi alle risorse non utilizzate sono da considerarsi sprechi di denaro per l'azienda.
- *Underprovisioning*: non sono state allocate risorse sufficienti per poter soddisfare tutte le richieste in arrivo, per questo alcuni utenti non potranno interagire con il sistema in maniera corretta. Gli effetti dal punto di vista economico causati da queste situazioni non sono spesso valutabili in maniera precisa, ma si possono rivelare molto importanti: non solo le richieste non soddisfatte non possono generare ricavi, ma vengono anche percepite come uno scarso livello di servizio da parte degli utenti, bisogna valutare infatti anche la possibile perdita di utenti causata dai disservizi dovuti a underprovisioning delle risorse.

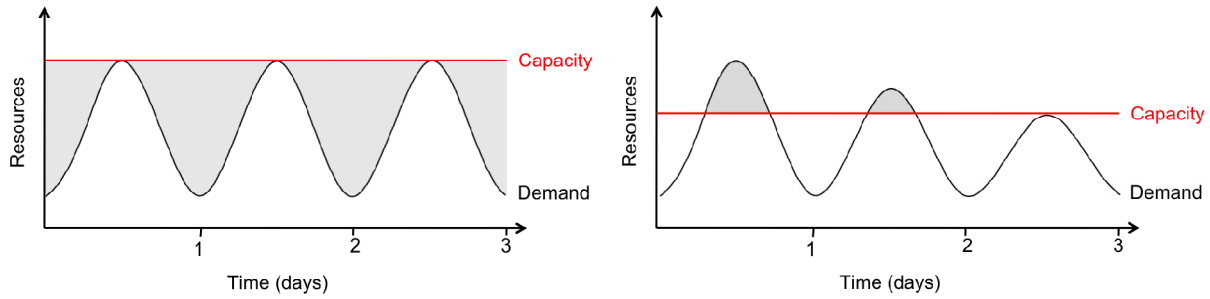


Figura 2 La variabilità delle richieste richiede di scegliere accuratamente la quantità di risorse da utilizzare, al fine di evitare fenomeni di *Overprovisioning* (grafico di sinistra) o *Underprovisioning* (grafico di destra)

Sfruttando l'elasticità di servizi offerti ogni azienda può ottimizzare la quantità di risorse allocata evitando di sovrastimare la quantità di risorse allocate e nello stesso momento avere le capacità di soddisfare carichi di lavoro maggiori rispetto alla norma. Visto che è possibile aumentare le risorse allocate in pochi minuti, i servizi IaaS sono molto indicati anche per le *startup*, le quali non conoscono a priori quanto i propri servizi verranno utilizzati e quindi il numero di risorse di cui avranno bisogno. L'utilizzo di servizi IaaS garantisce a tali attività le risorse necessarie per crescere anche in maniera molto importante, senza richiedere alcun tipo di garanzia o investimento preventivo, un esempio tipico di questa situazione è rappresentato da *Animoto* [20], che ha lanciato una propria applicazione su Facebook per creare degli slide show delle foto degli utenti: in soli tre giorni il numero di persone che utilizzavano il servizio di Animoto è passato da 25.000 a 250.000, richiedendo un aumento delle istanze utilizzate da 50 a 3.500. Se i server dell'applicazione fossero stati installati su macchine di un data center sicuramente l'aumento dell'utenza avrebbe saturato tutte le risorse, impedendo di soddisfare tutte le richieste pervenute.

2.3.2 Implementazione dell'IaaS – Virtualizzazione

Le tecnologie utilizzate per implementare i servizi IaaS si basano sui software di virtualizzazione, i quali in grado di poter gestire l'esecuzione di *macchine virtuali "guest"* su di una macchina fisica, detta host. In questo paragrafo verranno descritte brevemente le tre soluzioni utilizzate oggi per realizzare le funzionalità di virtualizzazione, alcune si basano su una serie di procedure implementate interamente tramite software, mentre altre sfruttano alcune funzionalità introdotte in hardware nei processori di nuove generazione [21].

- *Traduzione delle binary instructions*: il virtual monitor traduce le richieste kernel delle macchine virtuali in modo da sostituire le istruzioni non virtualizzabili con sequenze di istruzioni che abbiano il medesimo effetto sulla virtual machine; tutte le istruzioni eseguite a livello utente invece, vengono inviate direttamente al processore, in modo da massimizzare le performance. Ogni VM viene dotata di tutte le funzionalità caratteristiche di un sistema fisico reale, quali bios, periferiche e memoria. Questo tipo di virtualizzazione consente di astrarre totalmente la VM dall'hardware sottostante, il sistema guest infatti è inconsapevole di essere virtualizzato e di conseguenza non richiede nessun tipo di modifica preventiva.

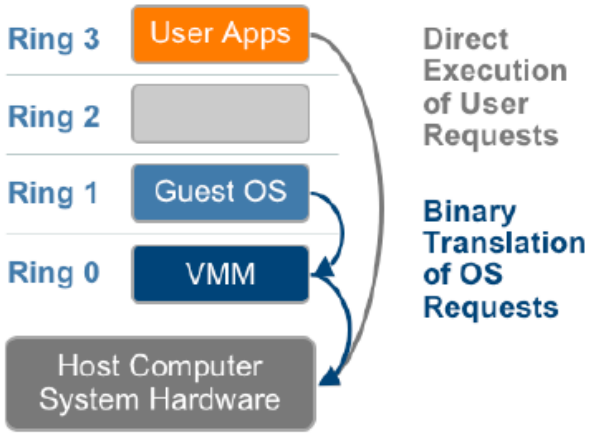


Figura 3 Schema di funzionamento per la Traduzione delle Binary Instructions

- *Paravirtualizzazione*: questo tipo di virtualizzazione si basa su un'interazione tra il sistema operativo Guest e l'hypervisor installato sulla macchina host, cercando di migliorare il più possibile le prestazioni del sistema. La Paravirtualizzazione richiede che il sistema operativo Guest si basi su un kernel modificato, in grado di sostituire le chiamate non-virtualizzabili con Hypercalls che vengono inviate direttamente all'hypervisor sull'Host. Dato che si richiede l'utilizzo esclusivo di sistemi modificati, questo metodo di virtualizzazione risulta poco portabile e scarsamente compatibile (non è infatti possibile installare OS non modificabili come Windows). È da sottolineare il fatto che prodotti basati su questo tipo di architettura (come Xen) risultino molto performanti e facilmente scalabili [22].

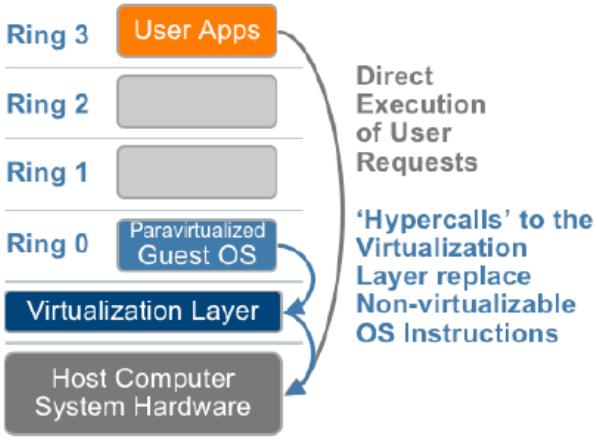


Figura 4 Schema di funzionamento della Paravirtualizzazione

- *Virtualizzazione mediante il supporto Hardware*: i produttori di CPU hanno introdotto recentemente (2006) alcune funzionalità sui propri processori per facilitare le tecniche di virtualizzazione. *Intel Virtualization Tecnology* (VT-x) e *AMD-V* consentono per alcune istruzioni un nuovo livello di esecuzione al di sotto del ring 0 (OS), accessibile dal Virtual Monitor. In questo modo tutte le richieste che dovevano essere tradotte o gestite dalla paravirtualizzazione, vengono intercettate e inviate direttamente alla CPU, che le esegue

normalmente, mantenendo tutte le informazioni dell'ambiente di esecuzione della macchina guest (quali registri, ecc). Questo sistema però non sempre consente di ridurre l'overhead di virtualizzazione, infatti ogni transizione dal Virtual Monitor alla VM (VMEntry) e vice-versa (VMExit), comporta un numero elevato di cicli di CPU, che va dalle centinaia alle migliaia di cicli [23].

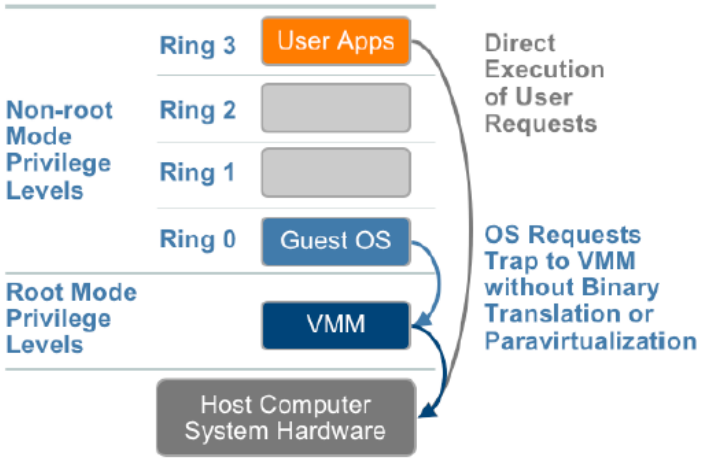


Figura 5 Schema di funzionamento della virtualizzazione utilizzando il supporto hardware

2.4 Problematiche che possono emergere nell'utilizzo di servizi cloud

Oltre ai vantaggi che si possono ottenere adottando i servizi di cloud computing delle diverse tipologie, si possono presentare una serie di ostacoli che tendono a limitare l'adozione dei sistemi cloud, sia per le aziende, che per i privati. Vi sono diverse tipologie di problematiche, a partire da quelle puramente tecniche, per arrivare a quelle più economiche.

Disponibilità del servizio

Gli utenti richiedono che i servizi offerti garantiscano un livello adeguato di disponibilità, infatti una volta che un privato o un'azienda decide di utilizzare servizi cloud, si aspetta che essi siano sempre disponibili e pronti all'uso, in certi casi infatti potrebbe risultare molto dannoso per il cliente se il servizio non fosse più accessibile. I *cloud provider* sono però molto attenti per cercare di mantenere la percentuale di uptime dei propri servizi più alta possibile, riuscendo a garantire degli ottimi livelli di disponibilità. Tuttavia, nonostante le tecnologie e gli strumenti utilizzati, può succedere che si verifichino dei malfunzionamenti, anche nel caso di service provider molto importanti: negli ultimi anni infatti alcuni malfunzionamenti che hanno reso indisponibili per diverse ore servizi molto famosi e utilizzati come quelli Gmail, Google App Engine [24] e Amazon S3 [25]. Quando si utilizza un servizio di cloud computing si deve considerare anche il rischio che questo diventi non utilizzabile per qualche lasso di tempo durante l'anno, di conseguenza è necessario valutare se sia fondamentale che questo resti invece attivo in maniera continuativa, se così fosse è obbligatorio optare per una scelta diversa. Bisogna sottolineare però che i cloud provider hanno sviluppato dei sistemi molto avanzati per poter garantire il massimo della disponibilità del servizio, in certi casi però si riescono ad ottenere garanzie di servizio

maggiori da un servizio IaaS rispetto a quanto si potrebbe ottenere utilizzando le risorse di un data center interno.

Lock-in

I sistemi che ogni *cloud provider* sviluppa per accedere ai propri servizi, quali API e interfacce web, sono quasi sempre proprietari e non-standardizzati. Solo ultimamente si stanno diffondendo alcuni standard definiti dall'accordo tra alcuni provider, tuttavia alcuni fornitori, come Amazon ad esempio, hanno di deciso di mantenere le interfacce proprietarie. Per questi motivi gli utenti potrebbero incontrare grandi difficoltà nell'interagire con diversi provider, in particolare se si richiedono delle migrazioni da un fornitore a un altro; si crea infatti un effetto di *lock-in* nei confronti del service provider attualmente utilizzato. Tali limitazioni possono sicuramente ostacolare alcuni utenti nell'adozione di servizi cloud, visto che il fatto di non poter cambiare fornitore senza difficoltà viene considerato molto negativamente. Dal punto di vista dei *cloud provider* invece, l'effetto *lock-in* si può rivelare un fattore positivo, visto che obbliga i clienti a mantenere lo stesso fornitore del servizio per evitare di sostenere ingenti spese di migrazione. Situazioni di questo tipo rendono gli utenti molto vulnerabili ad aumenti di prezzo imposti dal provider stesso, a problemi di affidabilità che potrebbero emergere e ad una eventuale chiusura del servizio (questa ultima situazione non è così improbabile, si è già presentata in alcuni casi, nei quali molti degli utenti hanno perso completamente i loro dati). I rischi descritti sono un fattore limitante, che deve essere considerato dagli utenti prima di decidere se utilizzare servizi cloud, è inoltre importante valutare in maniera preventiva quanto sia difficile e costosa un'eventuale migrazione dal service provider scelto ad un altro.

Trasferimento dei dati personali su server di terzi

Molti degli utenti sono riluttanti nel permettere a terzi di gestire tutti i loro archivi contenenti i propri dati personali o comunque dati di una certa riservatezza; a queste considerazioni puramente personali, si devono aggiungere anche le limitazioni che vi sono dal punto di vista legale. In alcuni stati infatti si richiede che tutti gli archivi contenenti dati sensibili siano protetti mediante sistemi che soddisfino certe condizioni o addirittura che i server contenenti tali informazioni siano collocati fisicamente all'interno dei confini dello stato, come l'USA PATRIOT Act., o all'interno degli stati della UE, come richiedono alcune norme europee. I *cloud provider* quindi si devono impegnare ad ottenere le certificazioni necessarie per poter gestire i dati personali dei propri clienti, diventando ad esempio HIPAA-compliant (Health and Human Services Health Insurance Portability and Accountability Act) [26].

Si deve considerare però che la maggioranza dei *cloud provider* fornisce oggi dei livelli di sicurezza sufficientemente avanzati per poter garantire ai propri utenti un'elevata riservatezza dei dati, in linea con la maggioranza delle soluzioni IT gestite in maniera autonoma dalle aziende. Molte delle tecnologie utilizzate, in particolare dai provider di servizi IaaS consentono di garantire agli utenti alti livelli di sicurezza, sfruttando tecnologie già consolidate quali lo storage di dati cifrati, l'utilizzo di VLAN, di firewall e packet filters per gestire le comunicazioni.

Trasferimento dei dati

Vi sono un certo tipo di applicazioni che stanno diventando sempre più *data-intensive*, come ad esempio quelle di business analysis o di elaborazione di contenuti multimediali. Si deve affrontare quindi il problema di dover trasferire le grandi quantità di dati che si devono elaborare. Il costo per trasferire un terabyte di dati per i diversi servizi cloud varia da 100$ a 150$, a seconda delle tariffe imposte dai cloud provider, e il tempo richiesto per l'invio tramite una connessione a banda larga di 20Mbit/s è decisamente elevato, dell'ordine di giorni. Una soluzione al problema proposto è quella di inviare i dati al cloud provider spedendo fisicamente gli hard-disk tramite corriere, visto che i tempi di trasferimento risultano comunque più brevi rispetto all'invio telematico, oltre a rivelarsi più convenienti dal punto di vista economico.

È evidente che tutte le applicazioni che richiedono un continuo scambio di dati con l'esterno siano poco adatte ad essere installate su cloud, tuttavia i provider spesso non fanno pagare le comunicazioni interne tra le istanze e i repository di dati, di conseguenza trasferire tutti i propri dati sui server remoti può risultare utile se si possono sfruttare le condizioni vantaggiose offerte per effettuare elaborazioni data-intensive recuperando i dati tramite collegamenti interni.

Prestazioni non prevedibili

Una delle caratteristiche peculiari dei servizi di cloud computing è quella di condividere le risorse disponibili nel data center tra i vari utenti, in modo da poter massimizzare l'utilizzo di queste e ridurre le tariffe applicate. Questo paradigma comporta però un discreto livello di incertezza sulle effettive *prestazioni* che un determinato servizio può garantire: nel caso di un servizio di IaaS, su un server fisico verranno avviate più macchine virtuali, appartenenti ad utenti diversi. In tali situazioni è possibile che le istanze virtualizzate interferiscano l'una sulle prestazioni dell'altra, soprattutto per quanto riguarda le operazioni di I/O o sui tempi di elaborazione delle CPU. Questo tipo di interferenze solitamente non creano particolari disagi per la maggior parte delle applicazioni, infatti vengono comunque garantiti dei valori prestazionali minimi. Molte di queste problematiche sono legate al tipo di hypervisor utilizzato per la gestione delle macchine virtuali, sono infatti questi programmi a gestire come le varie istanze debbano interagire tra loro.

Problemi legali e di comportamento

Nei sistemi di cloud computing può accadere che uno degli utenti utilizzi i servizi per fini poco corretti, se non illegali, questi comportamenti possono coinvolgere anche gli altri utenti del servizio. Infatti dal punto di vista di un nodo esterno alla cloud, tutte le comunicazioni che provengono dall'interno di essa risultano più o meno simili, è successo infatti che alcuni utenti abbiano utilizzato dei servizi IaaS per diffondere messaggi di posta indesiderata e, come conseguenza, i maggiori filtri antispam hanno introdotto nelle loro tabelle tutti gli ip del provider, bloccando anche i messaggi inviati da altri utenti [27]. Per risolvere questo problema i service provider hanno provveduto a definire dei regolamenti precisi in modo da evitare che si utilizzino i servizi in modo malevolo; un problema legato a questo

riguarda il trasferimento di responsabilità legale dal service provider all'utente, bisogna infatti che sia l'utente a dover rispondere dei propri comportamenti e non chi fornisce il servizio utilizzato.

Un ultimo problema che emerge, soprattutto per quanto riguarda i service provider IaaS, riguarda la gestione delle *licenze software*, infatti il modello pay-as-you-go non si adatta bene al normale sistema di vendita di queste, essendo basata sull'acquisto per ogni singola macchina. Una soluzione adottata oramai da vari provider IaaS è quella di stringere degli accordi con i produttori del software e poi di rendere disponibili delle istanze con tali sistemi già preinstallati facendo pagare un sovrapprezzo alla tariffa oraria standard. Ad esempio se si vuole avviare una macchina virtuale Windows tramite Amazon AWS il costo sarà di 0.15$ all'ora, rispetto ai 0.10$ per un'istanza dotata di un sistema operativo open source.

3. CLOUD COMPUTING IAAS

3.1 Applicazioni per il Cloud Computing IaaS

Alcune tipologie di applicazioni si adattano particolarmente ad essere sviluppate per l'esecuzione su piattaforme cloud, si tratta principalmente di tutte quelle applicazioni che hanno determinate caratteristiche sia per quanto riguarda le richieste di risorse, sia per la tipologia di utilizzo da parte degli utenti. Una delle maggiori potenzialità dell'IaaS è di essere elastico e di garantire la possibilità alle applicazioni di scalare facilmente, mettendo a disposizione una quantità di risorse virtualmente illimitata, per questo un'applicazione trarrà il massimo del vantaggio se le risorse di cui ha bisogno sono abbastanza *variabili*, o comunque poco prevedibili nel tempo. Una delle situazioni più rappresentative di questo tipo di situazione riguarda le applicazioni che gestiscono servizi legati ai social network, che in certi casi hanno registrato forti aumenti dell'utenza, e di conseguenza delle risorse computazionali richieste, molto rapidi e improvvisi. Per soddisfare tali richieste la soluzione IaaS è decisamente la migliore, visto che è possibile aggiungere risorse in tempi brevissimi e a discrezione dell'utente. Inoltre non è neppure necessario effettuare grandi investimenti iniziali per l'avvio del sistema, infatti le spese da sostenere saranno proporzionate alla quantità di risorse utilizzate, che a loro volta sono collegati al numero di richieste che il sistema deve elaborare.

I servizi di IaaS possono anche essere utilizzati per implementare applicazioni che non richiedono di essere rese disponibili agli utenti, o più in generale di essere eseguite, per periodi di tempo molto lunghi; casi di questo tipo possono accadere sostanzialmente per soddisfare dei *picchi di richieste* che avvengono in concomitanza di eventi mediatici importanti, come eventi sportivi, oppure all'esecuzione di procedure non standard in un'azienda, come può essere il rilascio di una nuova versione di un software o di un aggiornamento. In questi casi si richiede spesso di dover disporre di molte risorse IT per poter soddisfare tutte le richieste in arrivo, tuttavia se si volesse fare un upgrade del data center interno affinchè possa gestire picchi di richieste di tale intensità si dovrebbero affrontare degli investimenti molto importanti, che risulteranno poi poco sfruttati una volta terminata la fase di massimo utilizzo. Situazioni del genere si presentano anche quando un'azienda deve effettuare delle operazioni interne per effettuare delle modifiche al proprio sistema IT, dovendo elaborare grandi quantità di dati,

operazioni che richiederebbero molto tempo e risorse se dovessero essere eseguite internamente. Una soluzione che può essere adottata è quindi quella di utilizzare delle macchine virtuali IaaS dedicate a svolgere queste operazioni, riuscendo quindi a completare le procedure richieste in tempi più brevi e senza sovraccaricare il sistema; il vantaggio principale però è quello di non dover fare investimenti di nessun tipo per ottenere le risorse IT di cui si necessita, si dovranno solo pagare le tariffe del provider IaaS in base al tempo-macchina effettivamente utilizzato.

Web Applications

L'utilizzo più diffuso del Cloud Computing IaaS, e ad oggi quello che più vi si adatta, è costituito dalle *applicazioni web*. Le necessità che hanno tali applicazioni sono estremamente compatibili con tutti i servizi forniti dai provider IaaS: elasticità nell'allocazione delle risorse, tempi di risposta alle richieste brevi, connettività a banda larga efficiente, sistema di networking personalizzabile, ecc. La caratteristica più importante però riguarda la scalabilità intrinseca delle web applications, nella maggior parte dei casi è sufficiente replicare i diversi componenti (webservers, database servers, ecc) per aumentare il carico di richieste che l'applicazione può soddisfare.

Le applicazioni web sono spesso soggette a carichi di lavoro non costanti, con picchi che si discostano in maniera molto significativa rispetto alla media; l'elasticità tipica del cloud computing consente di allocare, in alcuni casi in maniera completamente automatica, risorse aggiuntive per sopportare carichi di lavoro elevati e poi di ridurre il numero di macchine una volta che il picco di richieste è terminato. Tutto questo è abbinato al modello di fatturazione a consumo, ideale anche per i casi di start-up, che possono attivare i propri servizi a basso prezzo, potendo comunque scalare a fronte di un aumento dell'utenza.

Testing di Applicazioni

Utilizzare un'infrastruttura cloud-based come *piattaforma di test* è un sistema largamente diffuso, l'obiettivo è solitamente quello di verificare che le applicazioni sviluppate siano sufficientemente scalabili in casi reali. Si possono infatti implementare delle piattaforme di testing di prestazioni per tutte quelle applicazioni che dovranno poi essere eseguite sui propri data center, al fine di verificarne le effettive potenzialità; i test di performance infatti potrebbero sovraccaricare eccessivamente il data center aziendale, compromettendo le funzionalità legate alle altre applicazioni. Utilizzando le risorse fornite dai provider IaaS si ha la possibilità di effettuare dei test in un ambiente totalmente separato, ad un costo molto basso, visto che la tariffazione è calcolata in base all'effettivo consumo di risorse. Ovviamente questo tipo di utilizzo di un'infrastruttura IaaS si applica solamente se l'architettura da utilizzare è compatibile con quella che impone il provider IaaS per le macchine virtuali fornite.

Applicazioni Mobile Interattive

Una tipologia di applicazioni che sta nascendo in questi ultimi anni è costituita da software sviluppati per i telefonini di nuova generazione, le quali devono essere in grado di rispondere in tempo reale alle richieste effettuate dagli utenti. Tale tipologia di applicazioni può trarre vantaggio dai sistemi cloud non

solo perché i servizi offerti devono essere sempre disponibili, ma anche perché basano le proprie elaborazioni su grandi insiemi di dati, che possono essere acceduti facilmente se gestiti dalla cloud stessa. Tutte queste applicazioni sono soggette a picchi di richieste molto imprevedibili, soprattutto perché si possono diffondere in tempi brevi, aumentato in maniera consistente il numero di utilizzatori. Il fatto di poter sviluppare tutti i servizi di questo genere utilizzando risorse di tipo IaaS garantisce ottimi livelli di disponibilità, oltre che la possibilità di scalare facilmente, qualora fosse necessario per soddisfare un aumento del numero di utenti.

Applicazioni Batch "Parallele"

Un'altra tipologia di applicazioni che ultimamente viene eseguita su piattaforme IaaS è costituita da tutti quei servizi che si occupano dell'*elaborazione intensiva di dati*, come gli strumenti di analisi che si utilizzando per effettuare operazioni di Business Intelligence. Anche in questo caso le applicazioni in questione necessitano di una quantità di risorse estremamente variabile, infatti molto spesso le richieste di elaborazione di dati vengono inviate sporadicamente e richiedono grandi capacità di calcolo durante la loro esecuzione, mentre una volta terminate queste operazioni il sistema non richiede molte risorse. Per questi motivi è utile poter usufruire di un servizio IaaS per eseguire le operazioni da svolgere, richiedendo le risorse durante le fasi di elaborazione, e rilasciandole quando l'applicazione ha completato il proprio lavoro. Un particolare utilizzo per applicazioni di calcolo intensive è quella della business analysis: ultimamente vengono utilizzate molte risorse per monitorare costantemente i comportamenti dei clienti, le loro abitudini di acquisto, la gestione delle supply chain, ecc. Le richieste computazionali di questi applicativi tendono sempre ad aumentare, visto che i dati da elaborare sono in continua crescita, inoltre le operazioni che devono svolgere non richiedono un utilizzo costante delle risorse informatiche, ma solamente quando l'utente richiede di visualizzare i dati aggiornati.

Applicazioni di questo genere spesso sono sviluppate per analizzare grandi quantità di dati (dell'ordine delle centinaia di GB o più) e richiedono molte ore di computazione, essendo per lo più CPU-intensive. Nel caso, molto frequente, in cui sia presente un buon livello di parallelismo delle operazioni che devono essere svolte, l'utente può trarre vantaggio dal modello pay-as-you-go del Cloud Computing, visto che è possibile ottenere una grande quantità di risorse ed utilizzarle solo per il periodo di tempo, anche relativamente breve, necessario per l'esecuzione delle operazioni pianificate. In questo modo si possono ridurre di molto i tempi di esecuzione di tutte quelle procedure che, se svolte nel data center locale, avrebbero richiesto tempi di elaborazione molto più lunghi. Un esempio di tali applicazioni è rappresentato da un'operazione eseguita per il giornale *The Washington Post* [28] che ha utilizzato 200 macchine Cloud per convertire un intero database di documenti in un formato adatto ad essere utilizzato nel web. Lo sviluppo di applicazioni di questo tipo è facilitato da alcune piattaforme di programmazione quali MapReduce di Google e la sua versione opensource HAdoop [29], che consentono al programmatore di dedicarsi al solo sviluppo dell'applicazione, lasciando ad esse il compito della gestione della parallelizzazione delle operazioni.

Esecuzione di applicazioni Desktop CPU-intensive

È possibile utilizzare le risorse cloud anche per eseguire alcune operazioni avviate da programmi *desktop*, al fine di velocizzarne i tempi di risposta. Applicazioni quali Matlab e Mathematica sono già in grado di sfruttare il Cloud Computing per effettuare alcune operazioni particolarmente costose, mentre altre applicazioni utilizzano risorse esterne per il rendering 3D. La categoria di applicazioni appena descritta è ancora in fase di sviluppo, infatti non sono presenti molte soluzioni che utilizzano questa tipologia di calcolo remoto, tuttavia è possibile che in certi campi si possano ottenere dei risultati molto buoni. Un esempio di un servizio che è stato lanciato da qualche mese degli USA è *OnLive* [30], che consente agli utenti di giocare con videogames di ultima generazione senza dover disporre dell'hardware necessario all'esecuzione del gioco stesso, infatti tutte le operazioni di calcolo, grafico e non, vengono effettuate dai server di OnLive, che invieranno solamente le immagini risultanti, visualizzate direttamente sullo schermo dell'utente. Sicuramente servizi così strutturati possono essere molto validi e in certi casi possono migliorare di molto le prestazioni che si potrebbero ottenere eseguendo i propri programmi il locale, tuttavia l'usabilità di tali soluzioni è vincolata ad una disponibilità di banda sufficientemente elevata. È fondamentale infatti evitare di avere tempi di risposta troppo lunghi, in particolare quando le applicazioni richiedono un livello di interattività molto elevato, coma accade appunto per i videogames.

Diffusione del Cloud Computing

L'utilizzo di tecnologie cloud si sta diffondendo sempre di più anche all'interno delle aziende, come rivelano alcune ricerche di mercato[1]. Si è visto che circa il 75% delle compagnie stanno già utilizzando, o hanno in programma di utilizzare entro i prossimi tre anni, risorse fornite da sistemi cloud. I prodotti più utilizzati sono quelli di tipologia SaaS, come le applicazioni offerte da *Salesforce.com*, essendo sicuramente le più user friendly e le più facili da implementare; per quanto riguarda le offerte IaaS tra i più interessanti si trovano lo storage remoto e la capacità di calcolo on-demand.

In which of the following ways is your organization currently using cloud computing offerings?	**Percent**
Access to extra computing power on demand	16.2%
Running application using a software as a service (SaaS) model	44.3%
Storage	23.4%
Other, please specify	3.8%
Not applicable - do not currently use any cloud computing offerings	46.8%

Tabella 1 Utilizzi dei servizi cloud

Le applicazioni più gettonate per essere eseguite su piattaforme cloud sono applicazioni collaborative (wiki,web conferencing) , applicazioni web e software di progettazione, ma anche l'utilizzo di storage on-demand. In particolare le aziende preferiscono eseguire su cloud tutte applicazioni che devono

[1] Cloud Computing SURVEY Exclusive Research from CIO magazine , june 2009

essere facilmente accessibili, non critiche per il business dell'azienda e di supporto, in modo da ridurre al minimo i disagi nel caso ci fossero problemi con il provider cloud.

Please select the option which best describes your organization's usage or plans for the following on demand offerings	On the radar or actively researching	Currently using or implementing	Plan to use in next year	Plan to use 1 - 3 years	Plan to use 3 - 5 years	No plans to use
Application platforms & development software	20.0%	34.2%	6.2%	5.3%	2.2%	32.0%
Collaboration tools	17.8%	44.9%	7.1%	6.2%	1.8%	22.2%
Enterprise application software	18.3%	34.4%	8.0%	6.3%	1.3%	31.7%
Personal productivity software	17.3%	34.1%	6.8%	5.5%	1.8%	34.5%
Utilities/management software	17.6%	43.2%	9.0%	3.6%	0.5%	26.1%
Servers	21.1%	32.7%	8.1%	4.5%	2.2%	31.4%
Storage	22.2%	32.6%	8.6%	7.7%	2.7%	26.2%

Tabella 2 Progetti futuri per l'utilizzo di servizi cloud nelle aziende

3.2 Case Studies

In questa sezione verranno presentati alcuni casi di studio di risorse di tipo Infrastructure-as-a-Service, evidenziando come sono stati implementati i servizi offerti e quali sono i benefici che gli sviluppatori e gli utenti possono ottenere.

3.2.1 SmugSmug

SmugSmug [31]è un portale web di photosharing, in cui ogni utente può caricare le proprie foto e video, condividendoli con il resto della community; vengono forniti anche servizi aggiuntivi quali photoediting e strumenti per la creazione di slide show. Dal 2006 SmugSmug utilizza il servizio cloud IaaS di Amazon Simple Storage Service (S3) per lo storage dei dati, e Elastic Compute Cloud (EC2) per la loro elaborazione, come il dimensionamento, modifica, ecc. Le immagini e i video degli utenti vengono inviati direttamente ai server di Amazon, consentendo a SmugSmug di evitare i costi di gestione per lo storage di una quantità di dati così elevata, dell'ordine di centinaia di TeraByte. L'azienda ha stimato che l'utilizzo di questo servizio le ha fatto risparmiare ingenti quantità di denaro: circa $500,000 già dal primo anno di utilizzo. Per quanto riguarda le elaborazioni CPU intensive, SmugSmug utilizza un sistema autoscalabile di istanze EC2 [32], costituito da una macchina "*master*" che controlla le operazioni eseguite da varie macchine "*worker*" il cui numero varia in base alle richieste ricevute.

Idle CPU Time (percentage)				
Last min	Last 5 min	Last 15 min	Lst hour	Last Day
38.99%	25.35%	33.97%	44.05%	26.16%

Tabella 3 Idle time delle istanze attive, in percentuale

Ogni istanza worker, una volta avviata, contatta l'istanza master che le fornirà i parametri dell'elaborazione richiesta; tutti i job vengono organizzati mediante un sistema di queuing proprietario di SmugSmug. Il componente centrale non solo smista le diverse operazioni sui worker disponibili, ma anche ne monitora il carico, decidendo in maniera totalmente autonoma se avviare nuove istanze (in caso di sovraccarico) o se terminarne (se il numero di richieste fosse diminuito).

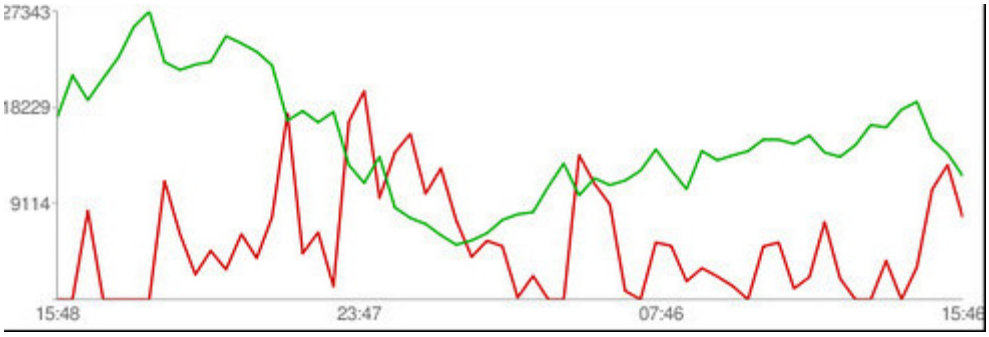


Figura 6 Idle vs Workers (24h)

I vantaggi di questo approccio sono molteplici: per prima cosa il sistema di autoscaling consente di utilizzare solamente le risorse di cui si ha bisogno in ogni momento, senza dover affrontare problematiche di overprovisioning, minimizzando anche in questo caso le spese; il sistema però si adatta in tempi brevissimi anche a soddisfare un grande numero di richieste, dato che in meno di 5 minuti è possibile avviare una nuova istanza "worker". Infine la fatturazione dei consumi avviene ogni 30 giorni, consentendo quindi di posticipare i pagamenti e di poter sfruttare appieno il proprio capitale disponibile. Gli amministratori di SmugSmug si dicono molto felici delle scelte fatte, soprattutto perché in questo modo possono dedicare tutte le loro risorse nel core business, senza doverne utilizzare per attività non strettamente legate al loro portale, quali appunto la gestione e il mantenimento di un data center così vasto da supportare il carico di lavoro richiesto dal portale

3.2.2 Phone Cloud

Un'applicazione possibile per sistemi di tipo IaaS è stata proposta da due membri dell'*NTT*[2] *Information Sharing Laboratory Group*, che hanno sviluppato un'architettura per sfruttare le risorse di calcolo cloud per migliorare le prestazioni di applicazioni per smartphone [33].

Il sistema proposto viene denominato *Virtual Smartphone over IP*, che consiste di un ambiente di cloud computing sviluppato appositamente per gli smartphone. Ogni utente può creare delle immagini virtuali del proprio dispositivo e eseguirvi da remoto le applicazioni che desidera, come se queste venissero utilizzate localmente. Il vantaggio principale di questo approccio è quello di superare le attuali limitazioni delle capacità di calcolo, memoria e soprattutto autonomia dei dispositivi mobili; eseguire le applicazioni su immagini remote consente anche di aumentare il livello di sicurezza, ad esempio evitando di dare accesso ai dati locali, creando una sorta di sandbox. Le applicazioni che vengono eseguite sulla cloud interagiscono con un apposito client installato sugli smartphone, il quale riceve le immagini di output (schermo) prodotte dall'istanza remota, mostrandole sul display dello smartphone. L'architettura del sistema è costituita da un *frontend*, attraverso il quale gli utenti possono avviare le proprie immagini virtuali e inoltrare le richieste di esecuzione delle applicazioni; è presente una *Virtual Smartphone Farm*, che contiene una serie di immagini già pronte all'utilizzo (ognuna dedicata a un singolo utente), di cui ne effettua la virtualizzazione quando richiesto. Per la gestione dei dati remoti

[2] Nippon Telegraph and Telephone Corporation

viene utilizzato un *Network File System* accessibile da ogni immagine virtuale; è presente infine un *management server* per la gestione della Virtual Smartphone Farm (creazione di nuove immagini, aggiornamento, deploy ecc.).

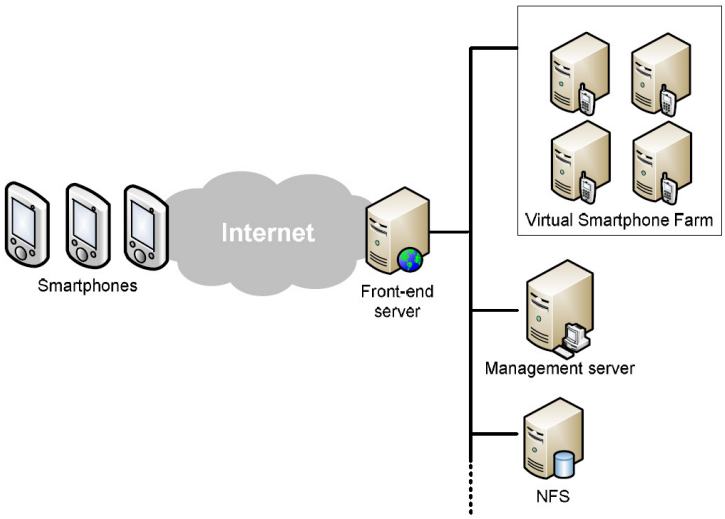


Figura 7 Architettura del sistema Phone Cloud

Si è verificato che se le operazioni eseguite richiedono elaborazioni abbastanza lunghe, o quantità di memoria sufficientemente elevate, l'utilizzo di una piattaforma remota migliora di molto le prestazioni del dispositivo mobile, sia per quanto riguarda i tempi di risposta, che per la durata della batteria, dato che il processore del dispositivo deve gestire solamente le comunicazioni e non elaborare i dati.

3.3 Opportunità e rischi del cloud computing IaaS per le imprese

È necessario valutare se una soluzione di cloud computing IaaS possa essere vantaggiosa non solo per piccole startup o aziende con particolari richieste, ma anche per quelle che richiedono risorse IT standard. Questa categoria di imprese solitamente gestisce un proprio piccolo data center interno, nel quale vengono eseguiti software commerciali o, in molti casi, applicativi personalizzati sviluppati ad-hoc. A questo punto ogni azienda potrebbe valutare l'idea di trasferire i propri server, tutti o alcuni di essi, su una piattaforma cloud, nel caso in cui questa migrazione risultasse vantaggiosa dal punto di vista economico e funzionale.

3.3.1 Modello per la valutazione della soluzione Cloud per le aziende

Klems e al. [34] hanno formalizzato una serie di operazioni che dovrebbe svolgere un'azienda per valutare se utilizzare un provider cloud per gestire le proprie risorse IT. La valutazione viene fatta in due passi principali: il primo in cui si definisce lo scenario in cui si sta lavorando e quali sono gli obiettivi da ottenere, il secondo in cui si effettua un analisi quantitativa per decidere se sia più conveniente una soluzione cloud o una diversa.

Per prima cosa si deve definire il proprio *Business Case* e gli obiettivi che si vogliono ottenere: si trovano diversi casi di studio che descrivono le varie aziende che hanno preso la decisione di utilizzare provider IaaS, tra queste sono presenti aziende di Application Hosting, Backup e storage, Content

delivery, E-Commerce, High performance Computing, Media hosting, On-Demand Workforce, Motori di ricerca e Web Hosting, alle quali si possono aggiungere delle nuove proposte come giochi online, servizi di update massivo e piattaforme per applicazioni dedicate ai social network. I benefici che si ricercano nel contesto del cloud computing sono una maggiore velocità di adattamento a domande di risorse molto variabili, alto livello di personalizzazione e tempi ridotti per entrare nel mercato. Ad esempio IBM sfrutta le risorse cloud per favorire lo sviluppo di processi innovativi all'interno dell'azienda, mentre l'U.S. Defence Information System Agency per velocizzare i tempi di sviluppo delle proprie applicazioni. Nel caso in cui i servizi che l'azienda vuole offrire siano simili a quelli citati sopra o possano trarre vantaggio dall'utilizzo della cloud, si devono valutare i benefici che si possono ottenere e come gestire le eventuali problematiche presenti. Uno dei primi problemi che si devono risolvere riguarda la *gestione remota dei dati*: innanzitutto si pone il problema della sicurezza degli stessi, che deve essere conforme alle politiche dell'azienda e non deve compromettere la fiducia dei propri clienti nel sistema che si sta utilizzando; un altro problema che si deve affrontare riguarda anche determinate leggi in vigore in alcuni paesi, che limitano l'utilizzo di database remoti per la gestione dei dati sensibili (spesso i dati devono essere mantenuti all'interno dei confini nazionali o dell'Unione Europea). Infine si devono valutare i rischi provocati da possibili effetti di *lock-in* futuri e dalla dipendenza dalle politiche di prezzi applicate del servizio di hosting; tuttavia i servizi cloud IaaS sono meno soggetti a queste problematiche, infatti è sicuramente più semplice migrare da una piattaforma IaaS ad un'altra (o a data center dedicati), piuttosto che migrare applicazioni sviluppate appositamente per piattaforme PaaS o SaaS.

Una volta verificato che una migrazione a un sistema cloud possa essere applicabile e che consenta all'azienda di migliorare la propria offerta, si passa alla valutazione quantitativa dello scenario proposto, al fine di verificare se questa scelta possa essere adeguata anche dal punto di vista economico. Conviene valutare se adottare una soluzione cloud o una tradizionale, verificando quanto i benefici offerti dall'IaaS siano importanti per l'azienda. Se si richiede esplicitamente che il sistema debba essere in grado di gestire un numero di richieste fortemente variabile e debba dare la possibilità di allocare nuove risorse in tempi molto brevi, la scelta deve obbligatoriamente andare nella direzione dell'IaaS, sia esso basato interamente su risorse fornite da un provider esterno, come Amazon EC2, o sia una cloud ibrida, in cui alcune risorse sono fornite da provider esterni, mentre altre sono risorse già presenti nel data center aziendale [35]. Se invece le applicazioni che si progetta di migrare sono soggette a richieste sufficientemente stabili e non richiedono una piattaforma fortemente dinamica, si deve valutare anche una soluzione di hosting dedicata tradizionale. Per valutare quantitativamente le differenze tra le diverse soluzioni vanno definiti degli attributi che indichino quanto i requisiti definiti nella prima fase vengano soddisfatti: questi di solito riguardano la capacità di storage, la potenza di calcolo richiesta o la quantità di banda consumata. A questi indici si devono aggiungere anche dei modificatori governati dai rischi e dalle opportunità che ognuna delle due soluzioni comporta, il metodo proposto per quantificare questi fattori è di assegnare ad ognuno un costo da aggiungere o togliere a

seconda dei casi. È sufficiente poi verificare quale delle due soluzioni sia più adeguata per ogni caso specifico e effettuare la scelta di conseguenza.

3.3.2 Case Study: migrazione di un sistema IT su IaaS

D.Greenwood et al. [36] hanno analizzato un caso particolare di una piccola-media impresa che produce soluzioni IT su misura per compagnie petrolifere, costituita da 30 dipendenti, con uffici in Inghilterra e in Medio Oriente. È stata analizzata la proposta di migrare uno dei prodotti più importanti dell'azienda, un servizio di monitoraggio e di acquisizione dati, su una piattaforma IaaS, in particolare Amazon EC2. L'applicazione viene eseguita su un data center interno composto da un database server, che riceve i dati da elaborare provenienti da sensori sulle piattaforme petrolifere, e da un application server che ospita diverse applicazioni di monitoraggio e controllo, che interagiscono con i clienti tramite Internet.

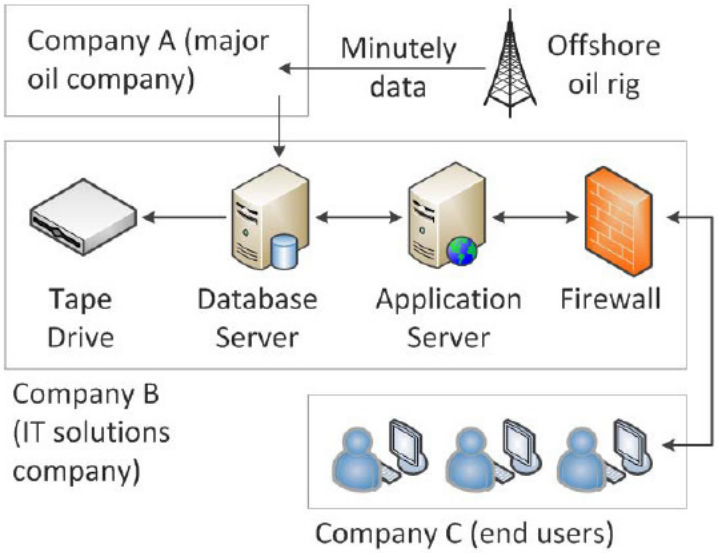


Figura 8 Architettura del sistema utilizzato dall'azienda prima della migrazione su cloud

Per prima cosa sono state analizzate le caratteristiche tecniche del sistema e valutati i costi necessari per la gestione e il mantenimento del data center, confrontandoli con le spese da sostenere per gestire un sistema analogo su Amazon EC2. Il costo della sola infrastruttura è stato quantificato in £19,400 per l'acquisto delle macchine, a cui va aggiunto un costo di £3,600 annuo per i consumi e il mantenimento, sommando il tutto per 5 anni di attività si ottiene un costo di £37,400.

La soluzione analoga proposta da Amazon EC2 è costituita da due macchine di tipo *small*, che possono essere sostituite con macchine *large* nel caso il carico di lavoro lo rendesse necessario. Questo è stato indicato già come un vantaggio nei confronti del sistema in-house, che invece non consente operazioni di "upgrade" così rapide. Sono stati valutati i costi di due istanze EC2 con sistema operativo Windows, operative per 730 ore mensili (attive 24h/24) con un traffico di 20GB in entrata e in uscita, utilizzando 200GB di spazio allocato su volumi EBS, 100 milioni di richieste di I/O, 30GB di spazio occupati da snapshot.

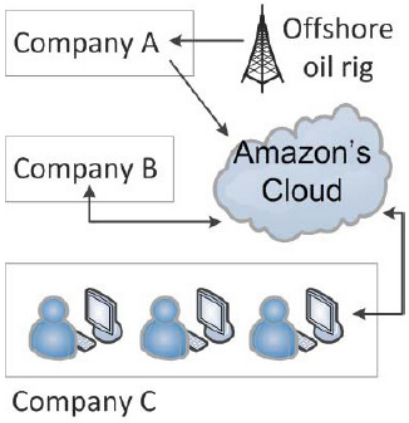


Figura 9 Architettura proposta con l'utilizzo del sistema cloud

Da un punto di vista dei soli costi, la soluzione di utilizzare un provider IaaS risulta essere la più conveniente, considerando il caso medio il risparmio entro 5 anni sarà del 37% rispetto alla gestione interna del data center. A questo vantaggio economico si deve aggiungere che per l'avvio del proprio data center remoto l'azienda non dovrà più effettuare alcun investimento preliminare, ma invece potrà pagare in base ai consumi effettuati ogni mese. La piattaforma IaaS consente anche all'azienda di avere l'opportunità di poter proporre nuovi servizi, senza doversi preoccupare di avere le risorse disponibili necessarie, basta infatti avviare una nuova macchina su EC2 per avere la potenza di calcolo richiesta; questa funzionalità, tipica delle piattaforme di IaaS, consente di avviare anche progetti sperimentali per un periodo di prova per poi decidere se mantenerli o meno ad un costo iniziale praticamente nullo.

Periodo	**Istanza Amazon**			**In-house Datacenter**
	2 small	**1 small +1 large**	**2 large**	
1 Mese	200	390	590	620
1 Anno	2400	4680	7080	7440
5 Anni	12000	23400	35400	37200

Figura 10 Costi calcolati per il mantenimento del sistema

La migrazione su IaaS influisce anche sui costi relativi al mantenimento del data center e alla gestione dei servizi di assistenza ai clienti. Il team di supporto dell'azienda ha il compito di monitorare lo stato del data center, effettuando dei controlli periodici sullo stato dell'hardware e del software, e di risolvere le problematiche sollevate dai clienti (principalmente tramite richieste telefoniche). Analizzando le richieste di assistenza passate, si è visto che circa il 20% delle chiamate riguardava problemi hardware, causati da guasti al sistema di backup (18%), problemi di networking (2%) o mancanza di corrente elettrica (1%); il restante 79% delle richieste riguardava invece problemi software. Nel momento in cui il data center venisse spostato su cloud, il 20% delle richieste di supporto verrebbe inoltrato direttamente ad Amazon, consentendo all'azienda di utilizzare le risorse risparmiate per altri obiettivi, più produttivi per il core business dell'azienda.

Dal punto di vista economico e tecnico la migrazione del sistema è sicuramente vantaggiosa, tuttavia sono stati considerati anche altri fattori, in particolare riguardanti la percezione di questa migrazione da parte dei clienti e dei dipendenti dell'azienda. Sono state effettuate delle interviste con manager e i principali azionisti dell'azienda, per evidenziare le problematiche che si potrebbero incontrare dopo aver effettuato la migrazione. Uno dei rischi maggiori tra quelli rilevati è quello di ottenere una riduzione del livello di qualità del servizio e della customer care, essendo dipendenti da un servizio gestito da terzi, soprattutto per richieste di interventi hardware. Inoltre il servizio di customer care potrebbe risultare meno efficiente visto che in certi casi è necessario risolvere le richieste dei clienti in cooperazione con il fornitore IaaS.

Questo caso di studio evidenzia come non si debbano considerare solamente i vantaggi economici di una migrazione, che per la maggior parte dei casi si rivelano favorevoli, ma è necessario verificare che il proprio business sia in grado di superare i rischi legati all'utilizzo di piattaforme IaaS e a gestire le problematiche che esse comportano.

3.3.3 Considerazioni

Si è visto quindi che nonostante tutti i vantaggi che una soluzione di tipo IaaS possa offrire, ogni azienda debba valutare attentamente se sia vantaggioso utilizzare servizi di cloud computing, infatti tali sistemi richiedono spesso una riorganizzazione di alcune delle soluzioni IT utilizzate. Non sono da sottovalutare anche i *costi operativi* da sostenere per l'utilizzo di una soluzione cloud, infatti non è detto che sfruttare le risorse fornite da un provider IaaS sia così vantaggioso dal punto di vista economico, soprattutto quando le risorse richieste non sono troppo elevate e variabili; la valutazione delle spese però non è così semplice da effettuare, come già accennato, infatti non si devono considerare solamente i costi effettivi dell'esecuzione delle macchine virtuali con quello di acquisto delle macchine fisiche, ma si devono valutare anche altri fattori. Per prima cosa la gestione di un data center richiede di sostenere delle spese relative alla manutenzione e all'utilizzo delle macchine, infatti si devono considerare i costi ordinari di locazione, come consumi energetici per l'alimentazione e la climatizzazione, e i costi straordinari che si devono sostenere per la manutenzione in caso di malfunzionamenti nell'hardware, come la sostituzione di componenti guasti. Oltre alle spese materiali bisogna anche considerare la quantità di risorse umane che deve essere dedicata al mantenimento del data center e di quanto questa possa essere diminuita se si adottasse usa soluzione IaaS.

Ogni azienda deve valutare anche come i propri clienti possano essere coinvolti nella migrazione su una piattaforma cloud; è necessario considerare come i servizi offerti potrebbero migliorare e di quali eventuali nuove funzionalità possono essere implementate successivamente. Un altro aspetto molto importante riguarda la percezione che i clienti possono avere di questa migrazione, infatti in certi casi il fatto di trasferire tutti i dati su piattaforme informatiche gestite da terzi non viene considerata sufficientemente sicura, per motivi di privacy o di garanzie di disponibilità del servizio; l'azienda quindi deve valutare le garanzie che vengono offerte dal provider IaaS scelto e decidere se queste

possano essere sufficienti a garantire un servizio efficiente ai propri clienti. L'utilizzo di una piattaforma IaaS può fornire all'azienda nuove possibilità di sviluppo, non attuabili se il sistema informatico fosse gestito in maniera tradizionale, infatti può essere utile sfruttare l'elasticità dei servizi cloud per fornire ai clienti funzionalità aggiuntive. Dato che non si richiedono investimenti aggiuntivi per allocare nuove risorse, ma le spese da sostenere sono solamente quelle relative al consumo effettuato, si possono rilasciare delle nuove applicazioni, che potranno essere eliminate se non riscontrassero il gradimento da parte dei clienti. Così facendo è possibile mettere in produzione nuovi servizi per un periodo di prova, senza dover affrontare ingenti spese aggiuntive, come ad esempio l'acquisto di una nuova macchina in un data center; se il nuovo servizio non riscontrasse il successo desiderato, è sufficiente deallocare le risorse destinate a quell'applicazione, senza aver effettuato alcun investimento inadeguato.

In generale avere il porprio sistema informatico, o parte di esso, su IaaS consente un grado di libertà molto maggiore rispetto ad un data center classico, visto che l'allocazione di risorse è immediata e non richiede alcun investimento aggiuntivo, ma solamente il costo di utilizzo; non si presenta quindi nessun tipo di problema riguardante la scarsità di risorse informatiche, consentendo agli sviluppatori di rilasciare tutti i servizi che ritengono opportuni per migliorare l'offerta ai propri clienti.

3.4 Perché diventare un provider IaaS?

Il cloud computing offre non solo opportunità per gli utenti dei servizi, ma presenta anche delle possibilità di crescita e di guadagno per tutte quelle imprese che dispongono delle conoscenze e delle risorse necessarie per poter fornire loro stesse un servizio *di cloud computing*; si deve notare però che solamente alcune compagnie possono scegliere di intraprendere un percorso di questo tipo.

Innanzitutto per poter fornire un servizio di Cloud Computing IaaS sono necessarie infrastrutture informatiche molto grandi e costose, infatti si richiede la presenza di una quantità sufficientemente elevata di risorse disponibili, quali CPU, spazio su disco e memoria, in modo da averne sempre disponibili per soddisfare le richieste inoltrate dai clienti. Oltre a questa richiesta di materiali è anche necessario possedere delle conoscenze molto avanzate nell'ambito della gestione di data center di così grandi dimensioni, si necessita di conseguenza anche di personale specializzato in grado di risolvere tutte le problematiche che emergono nella gestione di tali centri di calcolo. Per questi motivi molti dei provider di cloud computing di maggior successo hanno iniziato ad offrire i propri servizi cloud non come core business, ma solamente dopo aver già sviluppato le conoscenze necessarie nel campo, solitamente per gestire internamente i propri, già grandi, data center; la scelta di diventare cloud provider nella maggior parte dei casi è stata fatta a posteriori, dopo aver valutato di avere tutti i requisiti necessari per offrire un servizio sicuro e performante. In molti casi sono state sviluppate anche soluzioni software ad-hoc, le quali si sono poi rivelate molto utili anche per risolvere le problematiche che emergono durante la gestione di una piattaforma IaaS. Tra i più importanti fornitori di servizi cloud

oggi si trovano molte delle compagnie più grandi e affermate del settore IT, quali Amazon AWS, Microsoft Azure, IBM Cloud e Google App Engine.

Una volta verificato di essere in grado di poter gestire questo tipo di servizio, ogni azienda deve decidere come il diventare cloud provider possa essere vantaggioso per la propria particolare situazione; molti fattori possono influenzare tale scelta:

- *Possibilità di ottenere grandi guadagni:* dato che il prezzo di una VM si aggira intorno a 0.10$ all'ora, ogni cloud provider deve cercare di mantenere un numero molto elevato di clienti, e di riuscire a sfruttare il proprio potere contrattuale con i fornitori di hardware, software e banda in modo da ottenere prezzi sempre più bassi, così da massimizzare i propri guadagni. Anche i costi di gestione del data center vengono ammortizzati più facilmente quando le dimensioni di quest'ultimo sono sufficientemente grandi e se si hanno le conoscenze necessarie. Così facendo si potranno ottenere dei ricavi significativi, mantenendo le proprie spese sufficientemente basse e garantendo così ai clienti tariffe competitive.
- *Sfruttare investimenti già esistenti:* aggiungere un servizio di cloud providing su un'infrastruttura esistente consente di avere ricavi aggiuntivi a un costo sufficientemente basso, il che aiuta ad ammortizzare le spese sostenute per le risorse dei data center; ovviamente questo è applicabile solamente se si dispone di buone conoscenze tecniche nel settore. È frequente infatti che molte delle risorse di un data center risultino inutilizzate, in queste situazioni può risultare molto conveniente renderle disponibili per i servizi di cloud provisioning.
- *Difendere la propria posizione:* dato che molte applicazioni server e enterprise vengono sviluppate su piattaforme cloud, alcune aziende già presenti nel settore sono motivate a proporre anch'esse una soluzione analoga, per evitare di perdere la propria posizione nel settore. Ad esempio la piattaforma Cloud di Microsoft (Azure) propone un sistema diretto e semplice ai clienti di tipo enterprise di migrare il loro software sulla cloud.
- *Mantenere le relazioni con i clienti:* un provider già affermato, con delle relazioni contrattuali già stabilite con i propri clienti, può proporre un'offerta di Cloud Services, dando ad ogni cliente la possibilità di scegliere se e come utilizzare i servizi cloud, salvaguardando così gli investimenti sostenuti da entrambe le parti per la gestione del rapporto contrattuale.

Un'altra scelta strategica fondamentale riguarda il *collocamento geografico* del data center cloud, molte delle più grandi aziende che offrono servizi di cloud computing possiedono centri di calcolo localizzati in luoghi scelti accuratamente. Il criterio di scelta più importante è quello del costo dell'energia elettrica, infatti nei grandi data center le spese energetiche costituiscono oltre un terzo delle spese totali da sostenere. Conviene quindi posizionare i propri centri di calcolo in zone in cui è possibile ricevere energia elettrica a prezzi meno elevati. Oltre alla disponibilità di energia, un altro fattore che deve essere valutato è il costo della forza lavoro, è necessario infatti avere un numeroso team di persone per poter gestire sistemi così vasti; infine è fondamentale la vicinanza con i

fornitori di hardware e di banda, che permette di poter intervenire più velocemente per effettuare operazioni di manutenzione.

Location	Price per KWH	Possible Reason Why
Idaho	3.8 cent/$	Hydroelectric power; not sent long distance
California	10.0 cent/$	Electricity transmitted long distance over the grid; limited transmission lines in Bay Area; no coal fired electricity allowed in California.
Hawaii	18.0 cent/$	Must ship fuel to generate electricity

Tabella 4 Prezzi per KWH in alcuni stati USA

3.5 Provider IaaS

In questa sezione verranno presentati i più importanti provider di risorse cloud computing IaaS, tra questi sono presenti sia le soluzioni basate su hypervisor open source, che quelle che utilizzano i prodotti a pagamento di VMWare. Ogni provider fornisce tipologie di servizio particolari, ma in generale le offerte sono sempre costituite da macchine virtuali create on-demand in base alle richieste degli utenti e fatturate secondo il modello di pagamento a consumo (pay-as-you-go).

3.5.1 Amazon AWS

Amazon AWS [6] è sicuramente uno dei provider di servizi cloud più famoso e utilizzato dagli sviluppatori. I suoi servizi sono stati lanciati nel 2002, fornendo le funzionalità di base quali storage e capacità di calcolo on-demand, sono poi stati sviluppati servizi aggiuntivi al fine di migliorare l'offerta. I servizi di Amazon si possono dividere in due categorie: Infrastructure web services, che forniscono risorse fisiche a pagamento, e Application web services, che forniscono applicazioni a supporto dei clienti, ad esempio per gestire i pagamenti in un portale di e-commerce.

Infrastructure Web Services

I servizi di questa categoria sono stati sviluppati principalmente per i software developer, o più in generale, per tutte quelle aziende che hanno la necessità di avere dei sistemi informatici efficienti e scalabili; gli AWS si propongono quindi come *alternativa a basso costo*, alla costruzione e al mantenimento di grandi data center aziendali. I principali sono:

- *Simple Storage Service (S3)* lanciato nel marzo 2006, fornisce una semplice interfaccia attraverso la quale l'utente può salvare i propri dati sui server di Amazon, avendo poi la possibilità di accedervi in qualsiasi momento via web; il sistema crea automaticamente delle repliche dei dati distribuite geograficamente su tutta la rete di server, al fine di garantire la persistenza dei contenuti. Le tariffe non comprendono costi fissi mensili, bensì la quota da pagare viene calcolata in base alla quantità di dati che si mantiene sui server e al traffico (sia upload che download) generato; i prezzi vanno da $0.12 e $0.15 per GB salvato al mese, $0.10 per GB di traffico in upload e tra $0.10 e $0.17 per GB in download. I clienti di S3 sono sia privati che aziende emergenti, come SmugSmug (portale di photosharing), sia compagnie più grandi come Microsoft, che ha utilizzato questo servizio per la distribuzione di software ai propri clienti. Nel 2008 a questo servizio si è aggiunto *Amazon Elastic Block Storage*, un

servizio che consente di creare dei veri e propri volumi virtuali (dischi) che possono essere utilizzati per il salvataggio dei dati; questa funzionalità risulta molto utile se integrata con EC2, uno dei più importanti servizi forniti da Amazon.

- *Elastic Compute Cloud (EC2)* fornisce agli sviluppatori capacità di calcolo on-demand, scalabile secondo le esigenze del singolo cliente; la principale potenzialità di questo servizio è la possibilità di scegliere di volta in volta la potenza necessaria per il calcolo, per questo motivo è stata denominata "elastic". Gli sviluppatori che utilizzano questo servizio devono pagare in base al tempo-macchina occupato (a partire da $0.20 ogni ora), oltre a un corrispettivo simile ad S3 per i dati inviati e ricevuti. Sarà poi il cliente stesso che deciderà come utilizzare il servizio, avendo la possibilità di usufruire di molte risorse per un periodo breve di tempo, o poche risorse per un intervallo più lungo. La capacità di calcolo viene fornita dando all'utente il controllo di una o più macchine virtuali, che possono essere personalizzate secondo le singole esigenze, a partire dal sistema operativo utilizzato, fino ai programmi installati e alle configurazioni di rete dei dispositivi. È possibile interagire con il sistema mediante una suite di comandi, oppure tramite delle API REST o SOAP, utilizzabili tramite alcuni tool grafici come il plugin di Firefox chiamato ElasticFox.
- *SimpleDB* fornisce agli sviluppatori un database remoto, affidabile ed efficiente; offre la possibilità di effettuare query e lookup su dati strutturati. Il servizio è stato sviluppato per essere in grado di interagire con S3 ed EC2, fornendo ai clienti una piattaforma più completa. Anche un questo caso le tariffe sono calcolate in base al traffico generato (da $0.10 a $0.17 per GB trasferito) e allo spazio occupato ($0.25 al mese)
- *Simple Queue Service (SQS)* permette di inviare messaggi attraverso applicazioni distribuite, consentendo agli sviluppatori di evitare problemi dovuti alla mancata consegna di questi o alla gestione di tutte le varie eccezioni che potrebbero essere generate. Utilizzando questo servizio si rende molto più semplice la gestione del workflow di un'applicazione distribuita, in particolare quelle implementate su istanze di EC2. Anche in questo caso ogni cliente pagherà in base al traffico generato, con prezzi da $0.08 a $15 per GB trasferito.
- *Servizi Complementari:* Amazon oltre a questi servizi di base, fornisce anche altre funzionalità che consentono agli sviluppatori di sfruttare al meglio le potenzialità dei propri servizi di IaaS. Uno dei più importanti riguarda *Autoscaling* e *Amazon CloudWatch* il primo consente di creare dei cluster di macchine virtuali in grado di aumentare il loro numero e di diminuirlo in base al carico di lavoro richiesto in quel preciso momento; il secondo serve per monitorare lo stato delle macchine virtuali e dei dispositivi ad esse collegate, in modo da poter fornire informazioni ad Autoscaling per scegliere se e quando modificare il numero di istanze attive. Associato a questi, vi è *Elastic Load Balancing (ELB)*, un servizio che fornisce un load balancer in grado di gestire una quantità pressoché illimitata di richieste; è possibile collegare ELB ad Autoscaling,

riuscendo così a implementare un cluster di istanze in grado di auto dimensionarsi e di bilanciare il carico di lavoro su tutte le macchine virtuali attive.

Uno dei maggiori punti di forza di Amazon AWS è sicuramente il modello innovativo di fatturazione a consumo, che ha consentito a questi servizi di crescere e di essere utilizzati anche da una parte di clientela nuova, costituita da tutti quegli sviluppatori che non hanno la possibilità o la necessità di firmare contratti annuali con gli hosting provider, come ad esempio molte startup IT.

Prezzi

Amazon propone una scelta tra 8 tipologie diverse di VM, in base alle risorse assegnate, ossia quantità di Ram, numero di cores e spazio su disco allocato. Sono disponibili istanze dedicate ad applicazioni CPU intensive, mentre altre ad applicazioni che richiedono molta memoria. Ogni utente può scegliere quante risorse assegnare alle proprie VM, in base alle esigenze delle sue applicazioni e al denaro che è disposto a spendere; è possibile in qualsiasi momento effettuare un "upgrade" della macchina, spegnendo l'istanza in esecuzione e avviandone un'altra di tipo diverso. Oltre al costo relativo alle VM utilizzate, ogni utente dovrà pagare mensilmente in base a quanti dati mantiene in memoria (S3), quanto spazio ha allocato per i propri volumi (EBS) e la quantità di traffico che ha generato verso l'esterno. È da sottolineare che Amazon incentiva l'utilizzo delle proprie piattaforme di storage non fatturando i trasferimenti di dati interni da e verso le istanze, sfruttando questa possibilità molti clienti di Amazon possono utilizzare applicazioni che elaborano dati in maniera intensiva, memorizzandoli sfruttando le offerte di storage, il che consente di accedervi senza costi aggiuntivi (i trasferimenti di dati tra servizi dello stesso provider sono gratuiti).

	Costo per Macchina Virtuale (1 ora)			**Spazio Occupato (GB/mese) S3**	**Spazio Allocato (GB/mese) EBS**	**Traffico Generato (GB)**
Tipo di VM	Small (1CPU 1.7GB RAM)	Large (4CPU 7.5GB RAM)	Extra Large (8CPU 15GB RAM)			
Amazon EC2	0.035	0.34	0.68	0.15	0.10	0.15

Tabella 5 Tariffe applicate per i servizi Amazon EC2 (in $)

3.6 vCloud Express

VMWare, in collaborazione con alcuni partners, propone una soluzione di IaaS, denominata *vCloud Express* [37], anche in questo caso vengono messe a disposizione delle macchine virtuali personalizzabili dall'utente. La differenza fondamentale è rappresentata dell'hypervisor utilizzato e dagli altri applicativi messi a disposizione per realizzare l'infrastruttura, tutti basati su prodotti VMware, che verranno presentati più nel dettaglio nelle sezioni 6.5 e 6.6. I providers che forniscono i servizi vCloud Express sono: Bluelock, hosting.com, Melbourne IT e Terremark. Il sistema di pagamento utilizzato è quello del pay-as-you-go calcolato a seconda delle risorse consumate e del

traffico generato, con fatturazione mensile. Questi provider forniscono anche servizi di hosting tradizionali, per server virtuali o dedicati, con tariffazione mensile.

Bluelock Cloud

Bluelock [38] offre vari servizi di tipo IaaS, ognuno dedicato a diversi tipi di clientela:

- *vCloud Enterprise* – viene fornito un ambiente di cloud hosting personalizzato, con annessi servizi di management e gestione delle policy di sicurezza; questo servizio è dedicato alle medie-grandi imprese che richiedono delle soluzioni cloud complesse
- *vCloud Professional* – dedicato a tutte quelle aziende che necessitano di risorse IT gestite direttamente da Bluelock, viene garantita inoltre la sicurezza dei dati e la loro disponibilità
- *vCloud Express* – servizio IaaS on-demand, in cui ogni utente può creare le proprie VM e usufruirne secondo il modello pay-as-you-go simile ad EC2
- *virtual private cloud* – viene fornita una vera e propria cloud privata, con l'unica differenza che la gestione dell'hardware non deve essere fatta on-site ma sarà compito di Bluelock
- *Bluelock Hybrid Cloud* – mediante l'utilizzo di vCloud Director, i clienti potranno estendere le proprie risorse IT con quelle fornite da Bluelock, consentendo di spostare parte del carico di lavoro dalla propria cloud privata a quella di Bluelock

Hosting.com

Hosting.com [39] propone varie soluzioni, alcune basate su contratti a annuali, mentre altre con pagamenti a consumo.

- *Cloud VPS e Cloud Enterprise*, sono servizi che forniscono una o più macchine virtuali all'utente, il quale ha la possibilità di scegliere quale tipo di risorse assegnare ad ognuna di esse (RAM, cores, ecc.), con modalità di pagamento mensili calcolate in base alla quantità di risorse utilizzate. Il servizio Enterprise consente un maggior grado di personalizzazione, sia per la configurazione delle macchine, che per le policy di sicurezza.
- *Cloud Dedicated e Cloud private*, forniscono l'hosting di macchine fisiche dedicate per lo sviluppo di una cloud privata, che il cliente potrà gestire secondo le proprie esigenze. La versione Private aggiunge alcune funzionalità quali la possibilità di ottenere risorse aggiuntive on-demand. Il sistema di pagamento anche in questo caso è costituito da fatture mensili, calcolate in base alla quantità di risorse utilizzate.
- *vCloud Express (beta),* è il nuovo servizio di IaaS che consente di ottenere on-demand una quantità di risorse illimitate, utilizzando il modello pay-as-you-go, con prezzi calcolati in base alle risorse assegnate ad ogni VM e al traffico generato verso l'esterno.

Melbourne IT

Propone un servizio IaaS (ancora in beta) per i clienti residenti in Australia, il modello utilizzato è sempre incentrato sulla fatturazione a consumo. Ogni cliente può gestire direttamente le risorse richieste e richiederne di aggiuntive in ogni momento.

Terremark

Il servizio [40] propone un servizio di IaaS semplice, dando la possibilità agli utenti di creare fino ad un massimo di 60 VM, ciascuna con caratteristiche personalizzabili sia dal punto di vista dell'hardware (numero di CPU e quantità di RAM) sia per il sistema operativo usato (Linux o Windows). I Prezzi vengono calcolati in base ai tempi di uptime delle VM, allo spazio su disco occupato dalla immagini dei dischi e dalla quantità di dati trasferiti verso l'esterno. L'utente può gestire le proprie VM direttamente tramite l'interfaccia web o le vCloud API se volesse utilizzare client diversi.

Confronto Tariffe IaaS

I prezzi per l'utilizzo dei servizi offerti sono molto simili, in particolare per le risorse assegnate alle macchine virtuali (vedi Tabella). Ci sono invece alcune differenze per l'utilizzo di VM con sistemi operativi con licenza: Bluelock richiede un pagamento mensile di 33$ per ogni VM windows, mentre Terremark applica una tariffa oraria maggiore (di circa 0.010$); Melbourne IT e Hosting.com non supportano ancora sistemi operativi Microsoft.

	Costo per Macchina Virtuale (1 ora)									**Spazio Occupato**	**Traffico Generato**
RAM	512MB			1GB			4GB			**(GB/mese)**	**(GB)**
CORES	1	2	4	1	2	4	1	2	4		
Bluelock	0.033	0.038	0.044	0.058	0.068	0.078	0.229	0.269	0.295	0.35	0.17
Hosting.com	0.042	0.045	0.049	0.068	0.076	0.084	0.224	0.256	0.272	0.50	0.17
Melbourne IT	0.051	0.056	-	0.081	0.093	-	0.302	0.350	-	0.30	0.96
Terremark	0.035	0.040	0.045	0.060	0.070	0.080	0.241	0.281	0.301	0.25	0.17

Tabella 6 Comparazione dei prezzi proposti dai provider vCloud Express, espressi in dollari USA

I prezzi sono simili a quelli applicati dal diretto concorrente Amazon EC2, anche se quest'ultimo consente una personalizzazione ridotta delle istanze, essendo queste vincolate a solo 8 configurazioni (per RAM, CPU e storage).

3.6.1 Seeweb

Seeweb [41] propone soluzioni di hosting basate su infrastruttura cloud, ogni cliente può configurare il proprio server, anche a runtime, scegliendo la quantità di risorse che devono essere allocate a seconda delle necessità dell'applicazione. Il sistema di pagamenti si basa sull'acquisto di Server Power Unit (SPU) "spendibili" a piacimento dall'utente, il costo di 100 SPU è di 10€ al mese. Vengono forniti due tipi di hosting cloud:

- *Cloud Server*: si tratta di un server virtuale dedicato, le cui caratteristiche sono completamente personalizzabili dall'utente, che ha la possibilità di scegliere quale sistema operativo utilizzare, la quantità di memoria, lo spazio su disco e il numero di processori da allocare. Sono personalizzabili anche i servizi aggiuntivi come la protezione firewall, il servizio di backup e di assistenza. Il costo di ogni server viene espresso in SPU, calcolate in base alla quantità di risorse richieste dall'utente. Il vantaggio di utilizzare questo tipo di hosting è quello di poter "costruire" il proprio server, adattandolo alle proprie esigenze, pagando solo per quelle risorse che sono state richieste; inoltre essendo un server virtualizzato, è possibile modificare le risorse allocate a tal server in caso di necessità.
- *Cloud Hosting*: è una soluzione di tipo PaaS, in cui Seeweb fornisce un servizio di hosting web in grado di aumentare o diminuire le risorse utilizzate in base alle richieste dell'utente. Sono disponibili tutti i servizi tipici offerti dalle soluzioni tradizionali, quali server di posta, SQL e application server (Tomcat, Jboss,Zope,Ruby on Rails,..), che l'utente può utilizzare secondo le proprie esigenze. La potenzialità di questo tipo di soluzione sta nel fatto che le risorse allocate non sono più limitate da quelle fisiche dei server sottostanti, ma possono essere aumentate in maniera virtualmente illimitata. L'utente quindi dovrà solo effettuare una richiesta per l'aumento di risorse, a cui corrisponderà un aumento delle SPU mensili da corrispondere a SeeWeb, ma non dovrà modificare in alcun modo la struttura del proprio software.

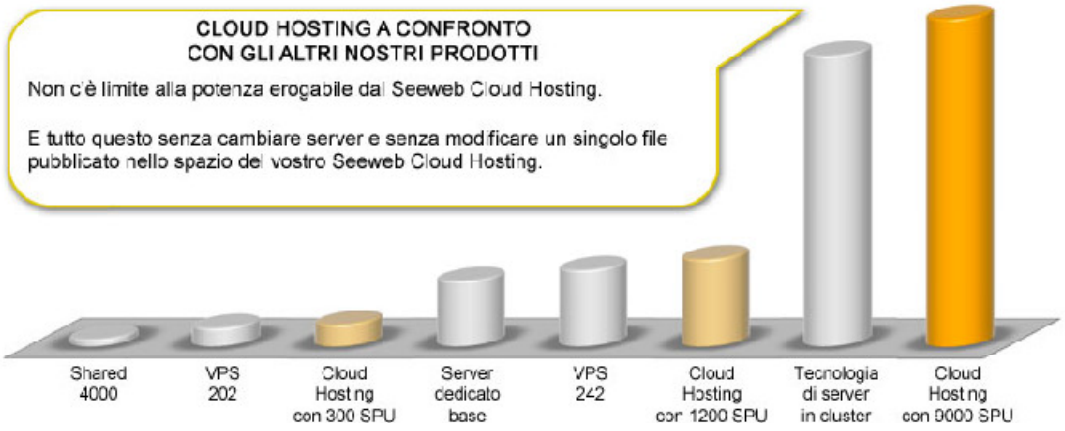


Figura 11 Quantità di risorse allocate per le diverse soluzioni di hosting proposte da Seeweb

Le soluzioni descritte utilizzano tecnologie di virtualizzazione basate su kvm, implementate su server fisici dotati di processori Xeon o Opteron (core >2.20GHz); tutta la topologia del data center è stata organizzata per essere ridondante e tollerante ai guasti, utilizzando un modello N+1. La disponibilità del servizio è garantita al 99.90%, con eventuali penali se tale percentuale non venisse rispettata.

3.6.2 GoGrid

GoGrid [42] fornisce servizi di IaaS, caratterizzati da una grande flessibilità e da buone possibilità di personalizzazione da parte dei cliente. Si possono attivare server virtuali, servizi di storage e networking, incluso un sistema di load balancing implementato in hardware, mediante l'utilizzo di API

standard. GoGrid garantisce 100% di uptime del sistema. Il sistema di pagamento considera le ore-RAM consumate da ciascuna VM che vengono addebitate mensilmente; un'ora RAM viene a costare da 0.10$ o meno, a seconda del consumo mensile effettuato. Per calcolare la tariffa applicata, viene fornito uno strumento utilizzabile mediante una semplice interfaccia web accessibile dal sito del provider. GoGrid fornisce anche servizi di hosting tradizionali, con fatturazione mensile a tariffa fissa; è possibile richiedere server virtuali oppure server dedicati.

3.6.3 Rackspace

Rackspace [43] fornisce servizi di hosting che comprendono, oltre a soluzioni tradizionali, anche servizi cloud: Cloud Server e Cloud Files.

Cloud Files è un servizio di storage remoto, che consente agli utenti di caricare file e pagare in base alla quantità di spazio che utilizzano, ad un prezzo di 0.15$ per GB al mese, a cui vanno aggiunti 0.22$ per ogni GB di dati trasmessi. Vengono fornite diverse interfacce utente grafiche, sia per PC che per altri dispositivi come gli smartphone, oltre che a delle API REST. La gestione interna dei dati è gestita in modo da replicare ognuno di essi più volte su macchine diverse, in varie zone dei data center. È anche possibile gestire gli accessi ai propri file, rendendone alcuni disponibili a tutti gli utenti.

Cloud Servers è un servizio di IaaS simile a quelli di vCloud Express, ogni utente infatti può scegliere le caratteristiche delle macchine virtuali che vuole utilizzare, pagando esclusivamente in base alle risorse richieste. Il calcolo del prezzo finale è basato anche in questo caso in base all'effettivo tempo di utilizzazione della macchina. I prezzi sono in linea con quelli degli altri provider (0.06$/ora per una VM con 1GB di RAM e 40GB di storage), viene applicato un sovraprezzo per l'utilizzo di macchine con sistemi Windows. L'infrastruttura sottostante è basata su un sistema di gestione IaaS sviluppato internamente, di cui verrà rilasciata una versione open source tra pochi mesi : OpenStack. Per la virtualizzazione viene utilizzato Xen hypervisor per le macchine linux e Xen server per quelle Windows, la persistenza fisica dei dati viene garantita da un sistema RAID-10 e ogni VM è dotata di 4 CPU virtuali; ogni Virtual Machine è dotata di due interfacce di rete, una per le comunicazioni interne (gratuite), e l'altra per quelle esterne (pagate a consumo). L'utente può interagire con il sistema mediante un'interfaccia web, oppure tramite delle API REST, utilizzabili da programmi personalizzati o di terze parti.

3.6.4 Confronto Prezzi IaaS

Il costo annuo di affitto di una VM fornita da provider di IaaS risulta essere superiore ad una macchina equivalente fornita con contratti tradizionali (sia essa una macchina dedicata o virtuale). Tuttavia non si può concludere che utilizzare i servizi di IaaS sia sempre meno conveniente, visto che le funzionalità offerte non sono le stesse. Inoltre la differenza di prezzo riscontrata tra un hosting tradizionale virtualizzato e la corrispondente soluzione IaaS non è così elevata; si può quindi valutare di scegliere un provider IaaS, considerando anche i servizi aggiuntivi offerti rispetto ad un aumento di prezzo spesso irrisorio. Per prima cosa bisogna considerare il tipo di applicazione che si vuole eseguire: nel caso si

richieda quantità di risorse molto variabili, la soluzione IaaS risulta sicuramente più conveniente rispetto alle altre; l'elasticità caratteristica di un sistema cloud si addice particolarmente a queste richieste, consentendo all'amministratore di non incorrere in errori di overprovisioning o underprovisioning. Si potrebbe supporre quindi che per applicazioni più costanti nell'utilizzo di risorse il sistema IaaS risulti in ogni caso poco conveniente e che sia preferibile utilizzare una forma di hosting diversa; spesso però i servizi offerti dai cloud provider possono essere comunque vantaggiosi, ad esempio per la loro elasticità nell'allocazione delle risorse, che consente di offrire nuovi servizi e funzionalità ai propri clienti senza dover effettuare investimenti di alcun tipo. Nel caso in cui un'azienda scelga di usufruire di servizi di hosting dedicati (virtuali o meno), dovrà dimensionare i server da affittare affinché tutte le applicazioni possano utilizzare le risorse di cui necessitano; si deve quindi valutare in maniera preventiva come strutturare il server affittato, inoltre solitamente si cerca di utilizzare server sufficientemente potenti per supportare anche eventuali picchi di richieste. Ne consegue che se si volesse migrare un'applicazione da Amazon EC2 a una piattaforma di hosting dedicato, sarà quasi sempre necessario utilizzare server più potenti rispetto a quelli che si utilizzano su EC2, visto che da una parte è possibile aumentare a piacimento la quantità di risorse richieste (in caso di un picco di lavoro), mentre dall'altra questo non è così semplice da attuare.

	Costo mensile	**Costo Annuo**	**Costo mensile storage (GB)**		**Costo Annuo Storage (GB)**		**Costo Mensile Banda GB**		**Costo Annuo Banda GB**		**Totale Annuo**
	1GB 1 core	1GB 1 core	100	500	100	500	100	500	100	500	1 core 1GB storage 100GB traffico 100GB
Bluelock	41.76	501.12	35	175	420	2100	17	85	204	1020	1125.12
Hosting.com	48.96	587.52	50	250	600	3000	17	85	204	1020	1391.52
Melbourne IT	58.32	699.84	30	150	360	1800	96	480	1152	5760	2211.84
Terremark	43.2	518.4	25	125	300	1500	17	85	204	1020	1022.4
Amazon Windows	122.4	1468.8	15	75	180	900	15	75	180	900	1828.8
Amazon Linux	61.2	734.4	15	75	180	900	15	75	180	900	1094.4
GoGrid	136.8	1641.6	15	75	180	900	29	145	348	1740	2169.6

Tabella 7 Costi annuali IaaS (in $)

Un'altra caratteristica che non viene considerata nella comparazione tra i costi riguarda le spese e i tempi richiesti per la manutenzione del sistema: nel caso in cui si utilizzasse un provider IaaS, se un'istanza incontrasse dei problemi di esecuzione o diventasse inutilizzabile, il problema potrà essere risolto dall'utente stesso avviando un'altra macchina analoga o addirittura da programmi esterni, che sono in grado di riconoscere i fallimenti delle VM e di gestirne il riavvio. Se invece lo stesso problema si presenta per un server dedicato, è necessario aprire dei ticket di assistenza, che devono essere risolti da personale incaricato, procedura sicuramente più lenta e costosa.

Hosting	Caratteristiche	Canone Mensile	Canone Annuo
Seeweb Cloud Hosting	1 core 1GB RAM 100GB storage 10Mb/s	110	1321
Seeweb Cloud Hosting	4 core 8GB RAM 100 GB storage 10Mb/s	-	2232
Seeweb Dedicated hosting	1 core 1GB RAM 160 GB storage 10Mb/s	-	2016
Aruba (server dedicato)	1 core 1GB RAM 160GB storage 10Mb/s	88	1060
GoGrid Dedicato	4 core 8GB RAM 360GB storage	-	2560

Tabella 8 Costi Annuali Hosting server dedicati (virtuali o fisici) (in $)

Un'azienda può anche valutare di acquistare fisicamente i server di cui necessita e di gestire il proprio data center internamente, soluzione ad oggi molto diffusa, soprattutto perché risultava l'unica applicabile fino a pochi anni fa. Se si considera solamente il costo per l'acquisto delle macchine, questa sembra di gran lunga la soluzione migliore: un server con le stesse caratteristiche di una macchina Amazon viene venduto a circa la metà del costo annuale per una VM. Bisogna però considerare che se si vuole effettuare questo tipo di scelta, si devono risolvere non solo le problematiche già esposte per i servizi di hosting dedicato, ma anche quelle legate alla gestione dell'hardware. L'azienda deve disporre di luoghi adatti al collocamento delle macchine, dotati di adeguati impianti elettrici e di condizionamento; inoltre non è sufficiente acquistare un singolo server per essere operativi, ma sono necessari anche dispositivi quali router, firewall o sistemi di backup di dati. Oltre a queste spese aggiuntive si deve considerare anche il costo del servizio di assistenza del produttore dei macchinari, che risulta d'obbligo per gestire i guasti. Infine si deve disporre di personale con conoscenze adeguate per svolgere mansioni di configurazione e gestione dell'intero sistema.

Ogni azienda deve quindi valutare accuratamente quali sono le proprie esigenze e quale soluzione risulti più vantaggiosa, sia in termini economici che per i servizi offerti. Le proposte cloud solitamente non richiedono spese aggiuntive rispetto ai prezzi di listino, mentre per la gestione interna si devono valutare anche i costi necessari per la gestione degli impianti. Non si può definire a priori se una soluzione sia migliore di un'altra, essendo tutte diverse tra loro, soprattutto per quanto riguarda la gestione dei dati, salvati in locale o in repository remoti, e per la gestione dell'hardware, mantenuto da terzi o da personale interno.

3.7 Migrazione da hosting privato a IaaS

Dopo avere effettuato le valutazioni necessarie per decidere di spostare il proprio data center su piattaforme IaaS, un'azienda deve affrontare le problematiche relative alla migrazione delle proprie macchine sulla cloud. Dato che servirsi di un provider IaaS non richiede investimenti preliminari, è conveniente che la migrazione del data center avvenga in maniera *graduale*, attivando di volta in volta le risorse necessarie. Una soluzione che può risultare vantaggiosa è quella di spostare prima i servizi più semplici, acquistando così l'esperienza necessaria per poter avviare la migrazione anche di quelli più complessi in una fase successiva.

Migrazione di un'applicazione

Effettuare la migrazione di un applicazione da un ambiente interno ad una piattaforma di cloud IaaS è un'operazione che può essere eseguita utilizzando diverse metodologie, in questa sezione verranno descritte alcune procedure standard, con riferimento particolare al provider di IaaS Amazon EC2.

Per migrare un'applicazione si possono seguire diversi percorsi: nel caso in cui si tratti di un'applicazione standard, come un database o un servizio web, senza particolari richieste di configurazione, conviene utilizzare una delle immagini che vengono messe a disposizione dal provider IaaS; su queste macchine di solito sono già installati gli applicativi necessari per l'esecuzione dell'applicazione, baserà esportare le proprie impostazioni personali e i dati per rendere il tutto operativo. Ultimamente alcuni dei fornitori dei software più utilizzati in ambiente server, come Oracle, Sun e IBM hanno reso disponibili delle immagini create appositamente per essere eseguite su Amazon EC2, nelle quali sono già integrati i loro software. Conviene quindi cercare di utilizzare questa tipologia di immagini, che garantiscono una compatibilità maggiore con il provider IaaS, visto che sono state sviluppate appositamente per essere eseguite su EC2 e molti dei parametri del sistema operativo e del software installato sono già stati ottimizzati per l'utilizzo nell'ambiente virtualizzato di Amazon. *Oracle* [44] ad esempio, ha reso disponibili immagini utilizzabili dai propri clienti utilizzando la stessa licenza che hanno già acquistato precedentemente, senza nessun costo aggiuntivo; i settaggi implementati sono stati studiati appositamente per sfruttare al massimo le potenzialità di EC2, consentendo ai clienti di Oracle di rendere scalabili e flessibili i propri sistemi. Seguendo questo schema implementativo si stima che la durata della migrazione possa ridursi da alcuni giorni a poche ore di lavoro. Se invece l'applicazione utilizzata è meno standardizzabile, si può installare il proprio applicativo su *un'immagine di base*, sempre fornita dal provider, configurare tutti i settaggi necessari e utilizzare dei tools per "salvare" la propria immagine, rendendola poi utilizzabile per l'esecuzione del software personalizzato. Anche in questa situazione conviene partire da un'immagine già predisposta per EC2, in modo da evitare incompatibilità che si possono riscontrare durante l'utilizzo di un sistema non correttamente configurato, infatti vi sono alcuni settaggi del sistema operativo che devono essere correttamente impostati, per rendere le istanze funzionanti.

Nel caso in cui un'applicazione sia fortemente dipendente dal sistema su cui viene eseguita, conviene effettuare una *migrazione dell'intera macchina* sulla piattaforma IaaS; Amazon fornisce una suite di comandi, ec2-tools, che permettono di convertire la propria macchina (virtuale su EC2 o locale) in un'immagine utilizzabile sulla propria cloud. I tool forniti da Amazon sono completamente compatibili con sistemi linux, è infatti possibile installare queste applicazioni e utilizzare il comando ec2-bundle-vol per creare un'Amazon Machine Image (AMI), un'immagine di una VM in formato compatibile con i servizi cloud di Amazon, della macchina in esecuzione; bisogna sottolineare che questo sistema richiede delle buone conoscenze dell'ambiente linux, tuttavia per uno staff IT sufficientemente esperto questo requisito non costituisce un ostacolo. Il comando ec2-bundle-vol è utilizzabile in maniera più diretta anche all'interno di una VM di EC2, consentendo di creare una nuova immagine esattamente

uguale a quella in esecuzione. Per quanto riguarda le immagini Windows, non è possibile effettuare l'upload di una propria immagine, più per un problema di gestione delle licenze d'uso che per problemi di compatibilità. Amazon propone, come nel caso linux, delle immagini windows già pronte per l'utilizzo, i cui prezzi orari sono leggermente superiori, appunto per pagare i diritti di licenza a Microsoft; se un utente desidera modificare un'immagine windows e salvarne l'immagine, con i tool Amazon può utilizzare il comando ec2-bundle-instance, il quale però non può essere lanciato direttamente dall'interno della VM, ma deve essere eseguito da un client esterno. Il sistema per prima cosa spegnerà la macchina virtuale windows e successivamente ne effettuerà il bundle e lo salverà in un bucket S3. Non è invece possibile effettuare una migrazione completa di una macchina fisica a un'immagine EC2, il suggerimento è quello di effettuare uno spostamento del singolo software, senza forzare una migrazione completa di tutta la macchina.

4. AMAZON EC2 – CAPACITÀ DI CALCOLO ON DEMAND

Come già citato in precedenza, Amazon EC2 è un servizio offerto da Amazon che consente agli utenti di avviare delle macchine virtuali on-demand, utilizzandole per eseguire le proprie applicazioni. Si devono però analizzare in maniera approfondita alcune caratteristiche tipiche di questo servizio web, visto che introduce dei cambiamenti abbastanza significativi dal punto di vista architetturale e funzionale. Ad un utente di Amazon EC2 infatti sono richieste delle conoscenze abbastanza specifiche nel campo linux, o Windows server, infatti molte operazioni da eseguire necessitano di essere configurate opportunamente con parametri specifici, spesso dovendo interagire con il sistema tramite lnea di comando. Oltre a queste conoscenze, ogni utente deve essere al corrente di come è strutturata l'offerta dei servizi Amazon, in modo di poter sfruttare al meglio tutte le diverse soluzioni che vengono proposte.

4.1 Caratteristiche di Amazon EC2

Il servizio *EC2*, oltre al modello di pagamento pay-as-you-go già citato, fornisce agli utenti una grande possibilità di personalizzazione di tutte le risorse fornite, si possono decidere infatti non solo la quantità delle macchine virtuali da attivare, ma anche quale tipo di immagine utilizzare e decidere dove questa dovrà essere allocata, inoltre si possono definire tutte una serie di parametri legati alla gestione della sicurezza delle comunicazioni tra le diverse istanze attive. Amazon infatti dispone di data center dislocati in diverse zone del mondo, dando così la possibilità agli utenti di scegliere la collocazione migliore per le loro istanze. Nei seguenti paragrafi verranno descritte nel dettaglio tutte le informazioni necessarie per poter utilizzare al meglio i servizi di Amazon, descrivendo anche un esempio pratico di utilizzo.

4.1.1 Tipi di Istanze

Per prima cosa ogni utente deve decidere quale tipo di istanza vuole avviare per eseguire le proprie applicazioni; il tipo di istanza definisce la quantità di risorse che Amazon allocherà per la macchina virtuale, che determina anche la tariffa oraria che verrà fatturata all'utente. Amazon non permette di impostare direttamente i parametri fisici delle istanze, come la quantità di memoria o lo spazio su disco,

ma fornisce una serie di modelli tra cui scegliere; questa limitazione non è però così negativa, visto che la gamma di modelli proposti è abbastanza ampia e consente di soddisfare le richieste di ogni tipo di utente.

Type	CPU	Memory	Local Storage	Platform	I/O	Name
Small	1 EC2 Compute Unit (1 virtual core with 1 EC2 Compute Unit)	1.7 GB	160 GB instance storage (150 GB plus 10 GB root partition)	32-bit	Moderate	m1.small
Large	4 EC2 Compute Units (2 virtual cores with 2 EC2 Compute Units each)	7.5 GB	850 GB instance storage (2 x 420 GB plus 10 GB root partition)	64-bit	High	m1.large
Extra Large	8 EC2 Compute Units (4 virtual cores with 2 EC2 Compute Units each)	15 GB	1690 GB instance storage (4 x 420 GB plus 10 GB root partition)	64-bit	High	m1.xlarge
Micro	Up to 2 EC2 Compute Units (for short periodic bursts)	613 MB	None (use Amazon EBS volumes for storage)	32-bit or 64-bit	Low	t1.micro
High-CPU Medium	5 EC2 Compute Units (2 virtual cores with 2.5 EC2 Compute Units each)	1.7 GB	350 GB instance storage (340 GB plus 10 GB root partition)	32-bit	Moderate	c1.medium
High-CPU Extra Large	20 EC2 Compute Units (8 virtual cores with 2.5 EC2 Compute Units each)	7 GB	1690 GB instance storage (4 x 420 GB plus 10 GB root partition)	64-bit	High	c1.xlarge
High-Memory Extra Large	6.5 EC2 Compute Units (2 virtual cores with 3.25 EC2 Compute Units each)	17.1 GB	420 GB instance storage (1 x 420 GB)	64-bit	Moderate	m2.xlarge
High-Memory Double Extra Large	13 EC2 Compute Units (4 virtual cores with 3.25 EC2 Compute Units each)	34.2 GB	850 GB instance storage (1 x 840 GB plus 10 GB root partition)	64-bit	High	m2.2xlarge
High-Memory Quadruple Extra Large	26 EC2 Compute Units (8 virtual cores with 3.25 EC2 Compute Units each)	68.4 GB	1690 GB instance storage (2 x 840 GB plus 10 GB root partition)	64-bit	High	m2.4xlarge
Cluster Compute	33.5 EC2 Compute Units (2 x Intel Xeon X5570, quad-core "Nehalem" architecture)	23 GB	1690 GB instance 64-bit storage (2 x 840 GB plus 10 GB root partition)	64-bit	Very high (10 Gbps Ethernet)	cc1.4xlarge

Tabella 9 Tipi di Istanze di EC2

I tipi di macchine proposti differiscono principalmente su quattro parametri:

- Numero di CPU
- Quantità di memoria
- Spazio su disco
- Prestazioni di I/O

Per poter adattarsi alle varie tipologie di applicazioni che si prestano ad essere eseguite da EC2, vengono proposte diverse tipologie di macchine divise per “famiglie” di applicazioni: standard, micro, High-CPU, High-Memory, Cluster Compute. Ognuna di esse è dedicata a un determinato tipo di applicazione, ad esempio le macchine della famiglia High-CPU sono dotate di una grande potenza di calcolo, dedicata appunto a tutte le applicazioni CPU-intensive, che magari richiedono una quantità di

ram non troppo elevata; in questo modo Amazon EC2 permette ai suoi utenti di pagare solamente per le risorse che gli sono effettivamente necessarie, senza dover spendere ulteriore denaro per altre non utilizzate.

La scelta della corretta configurazione della macchine risulta fondamentale, soprattutto nel caso in cui si debba utilizzare un cluster costituito da un numero di istanze abbastanza grande; effettuando una scelta appropriata si può risparmiare una quantità considerevole di denaro, senza dover sacrificare le prestazioni della propria applicazione. Bisogna sottolineare anche che non è necessario fare una scelta cercando di mantenere una certa percentuale di risorse libere per eventuali utilizzi futuri, come ad esempio si è soliti fare quando si acquista una macchina fisica. Se le prestazioni di un'istanza diventassero insufficienti, basta avviare una nuova macchina, utilizzando un modello con un maggior numero di risorse, e sostituirla a quella avviata precedentemente. Questo tipo di soluzione è reso possibile dal fatto che per avviare una nuova istanza è sufficiente inviare una richiesta su server di EC2, che in pochi minuti la renderanno attiva e funzionante, cosa impensabile se si utilizzasse un servizio di hosting tradizionale, in cui è necessario fare delle richieste specifiche, che possono essere eseguite con tempi dell'ordine di qualche giorno.

4.1.2 AMI – Amazon Machine Images

Una volta scelto il tipo di istanza che si vuole avviare, si deve decidere quale immagine utilizzare per l'esecuzione della macchina virtuale. Una Amazon Machine Image contiene tutte le informazioni che servono ad una virtual machine per essere avviata, come il sistema operativo e tutto il software necessario all'esecuzione dell'applicazione scelta. Amazon rende disponibili molte immagini pronte all'uso: sono presenti AMI di base su cui sono installati i sistemi operativi più utilizzati, a partire dalle più famose distribuzioni linux alle immagini Windows; sono disponibili anche AMI dotate di programmi preinstallati pronti all'uso, come ad esempio server MySQL o Oracle, o web server apache. Gli utenti possono scegliere se utilizzare tali immagini aggiungendo semplicemente i propri dati, oppure modificarle, creando delle proprie AMI personalizzate; tali immagini possono anche essere rese pubbliche, consentendo così ad altri utenti di utilizzarle per le loro attività. La scelta che si consiglia agli sviluppatori è quella di utilizzare, per quanto possibile, immagini pubbliche come base di partenza, e modificarle in piccola parte dove necessario, infatti tali AMI sono state create appositamente per sfruttare al massimo le potenzialità del sistema EC2, impostando i parametri del sistema operativo e delle applicazioni in modo da ottenere le prestazioni migliori. Qualora le immagini offerte nel catalogo di Amazon non fossero adatte alle applicazioni che si vogliono utilizzare, è possibile creare delle proprie immagini a partire da host reali o da immagini di macchine virtuali, che poi possono essere caricate sui server Amazon.

Immagini a Pagamento

Per poter eseguire determinate immagini, su cui è installato software protetto da copyright, si possono utilizzare delle AMI che richiedono un pagamento orario maggiorato, in modo da poter corrispondere i

diritti alla casa produttrice. Uno degli esempi più importanti riguarda sicuramente le immagini con sistema operativo Windows, disponibili tra quelle offerte per EC2, ma per poterle eseguire l'utente deve pagare una tariffa leggermente maggiore, di circa il 25%, a parità di risorse allocate.

4.1.3 Immagini EBS-Backed e S3-Backed

Uno degli aspetti più particolari di Amazon EC2 riguarda la gestione della persistenza dei dati delle immagini in esecuzione: si può scegliere di eseguire le proprie macchine virtuali in due modalità differenti: *EBS-backed* (disponibile da dicembre 2009) e *S3-backed*. La differenza sostanziale tra queste due modalità riguarda come vengono gestiti i dati prodotti durante l'esecuzione dell'istanza: in un caso questi restano memorizzati su un disco di EBS, mentre nell'altro vengono eliminati una volta che la macchina virtuale ha terminato la propria esecuzione.

S3 Backed

Le istanze lanciate da immagini S3-backed sono in realtà una copia dell'immagine che si è scelto di utilizzare: il sistema infatti, quando riceve una richiesta di avvio per una nuova istanza, crea una copia di tutti i dati necessari e li salva su un disco locale (virtuale) che verrà utilizzato come partizione di root. Da questo momento in poi, tutte le modifiche che verranno effettuate sull'immagine dell'istanza in esecuzione, come modifica dei dati, delle impostazioni del software ecc. saranno solo memorizzate localmente sul disco virtuale associato alla macchina. Nessuna modifica può essere fatta sull'immagine originale, che ha solamente il ruolo di stato di partenza, da cui vengono copiati i dati. Quando viene spenta un'istanza, tutti i dati contenuti sul disco vengono eliminati, così come lo stato del sistema; per mantenere in memoria i dati prodotti da una certa istanza si devono utilizzare tecniche di salvataggio dei dati diverse, come ad esempio collegare al sistema un disco di EBS, o utilizzare S3.

EBS-Backed

Un'istanza EBS-backed utilizza un disco di Elastic Block Storage come partizione di root, invece di utilizzare un disco virtuale, come per la soluzione precedente. Il funzionamento di questo tipo di istanze è molto simile a quello che si verifica quando si utilizza una macchina virtuale su un normale PC desktop, infatti tutti i dati dell'immagine della macchina sono salvati su un supporto persistente, di conseguenza non vi è nessun pericolo di perdita di dati dopo lo spegnimento della macchina virtuale. Le principali caratteristiche delle istanze di questa tipologia sono:

- La possibilità di allocare una maggior quantità di spazio per la *partizione di root* del sistema, limitata a 10GB per le istanze S3-backed; utilizzando un EBS invece si può montare una partizione fino a 1TB di capacità.
- *Maggior velocità in fase di boot*, visto che i dati del disco non devono essere completamente copiati prima di avviare l'istanza, ma basta accedere ai file necessari per il boot della macchina. Bisogna sottolineare però che in fase di runtime questo tipo di istanza risulta meno performante,

in quanto i dischi volatili delle altre istanze sono creati localmente sulla macchina host, mentre i volumi EBS vengono acceduti tramite connessioni a servizi esterni.

- *Possibilità di spegnere un'istanza*, che andrà nello stato "stopped" e riavviarla in un secondo momento, mantenendo lo stato del sistema e senza dover pagare le ore-macchina quando questa è spenta.
- Sono presenti delle *API* per creare delle EBS-backed AMI a partire da istanze in esecuzione, il processo risulta molto semplice, in quando la procedura consiste semplicemente nel creare un nuovo volume EBS, associarlo ad una configurazione di boot e registrarla.

Il vantaggio maggiore di queste istanze rispetto a quelle S3-backed è sicuramente il poter essere sospese e poi riattivate in momenti successivi, senza perdere lo stato del sistema; macchine di questo tipo si comportano, di fatto, come macchine reali.

Commenti

Non è detto che vi sia una scelta migliore tra immagini S3-backed e EBS-backed, ogni utente dovrà considerare la sua particolare situazione e valutare quale delle due soluzioni si migliore. Bisogna per prima cosa valutare il *costo* delle due differenti soluzioni, infatti utilizzare un'istanza di tipo S3-backed richiede solo di pagare il tempo effettivo nel quale è stata eseguita la macchina virtuale, senza nessun costo aggiuntivo per gli accessi sul disco locale. Se invece si utilizza un'istanza EBS-backed si devono considerare anche i costi dello storage dei dati su EBS, oltre che agli accessi effettuati durante l'esecuzione dell'istanza.

Vi è anche una differenza sensibile di *prestazioni* tra l'accesso ai dati su un EBS e su un disco locale: il primo infatti viene acceduto in maniera remota all'interno della rete di Amazon, si è visto infatti che le operazioni di I/O effettuate su dischi EBS sono meno veloci rispetto a quelle effettuate su dischi locali. Conviene quindi utilizzare soluzioni S3-backed se si richiede che gli accessi su disco siano molto veloci e frequenti.

Un altro scenario in cui l'utilizzo di immagini S3-backed risulta migliore si presenta quando si richiede di avviare una serie di istanze identiche, per poterle poi raggruppare in un *cluster* facendole collaborare tra loro. Una situazione del genere può trarre vantaggio dal sistema proposto da Amazon, infatti da una sola immagine "base" si possono attivare quante macchine virtuali si desidera. Uno dei vantaggi associato a questo tipo di implementazione riguarda la possibilità di separare il software, ossia il sistema operativo, con annesse applicazioni e impostazioni, dai dati utilizzati e prodotti dal software in esecuzione sulle istanze. Inoltre utilizzare un'organizzazione sistemistica del tipo appena descritto consente di ottenere dei livelli di fault-tolerance e di disaster recovery più alti, infatti se il volume EBS associato ad un'istanza EBS-backed dovesse corrompersi, tutti i dati dell'immagine verrebbero persi, mentre se si dovessero avere problemi con i dischi locali di un'istanza S3-backed si richiederebbe solamente un riavvio della macchina virtuale per rendere tutto di nuovo operativo.

Bisogna sottolineare però che l'utilizzo di immagini EBS-backed in certi casi risulta molto utile, se non obbligatorio, come nel caso delle immagini Windows Server, dato che le partizioni di root utilizzate per le immagini S3 sono di soli 10GB, troppo pochi per un'installazione del sistema operativo Microsoft. Ogni utente quindi dovrà decidere se nella sua situazione convenga gestire un grado di complessità maggiore, utilizzando immagini S3-backed, oppure mantenere lo stato di tutte le proprie macchine virtuali su dischi EBS.

4.1.4 Networking all'interno e all'esterno della rete EC2

Ad ogni istanza EC2 vengono assegnati, in fase di avvio, due indirizzi IP, uno privato e uno pubblico, collegati tra loro tramite un sistema di NAT; gli indirizzi privati sono raggiungibili solamente dall'interno della rete Amazon, mentre quelli pubblici sono accessibili dalla rete Internet. Viene offerto anche un servizio DNS che associa dei nomi mnemonici alle istanze, questo servizio risulta molto utile soprattutto per rendere accessibili le proprie istanze dall'esterno utilizzando una traduzione di un altro nome con quello dell'istanza EC2 associata (ad esempio si può associare il nome www.miosito.it ad un nome EC2 fornendo questi dati al server DNS competente).

Security Groups

Per gestire le policy di sicurezza della rete tra le istanze, si possono assegnare le macchine virtuali a dei *security groups* definiti dall'utente, il quale può personalizzare le regole del firewall da applicare in ogni gruppo. Tali regole non sono relative ad una singola istanza, ma vengono applicate all'intero insieme di istanze: si facilitano cos' tutte le operazioni di modifica e aggiunta di nuove regole, che verranno automaticamente applicate a tutte le istanze appartenenti a quel security group. Le regole che si possono definire in un security group sono molto simili a quelle disponibili per l'applicazione *iptables* nei sistemi Linux. La divisione in gruppi delle istanze consente inoltre di creare degli insiemi di macchine virtuali totalmente isolati tra loro, garantendo quindi un ottimo livello di sicurezza.

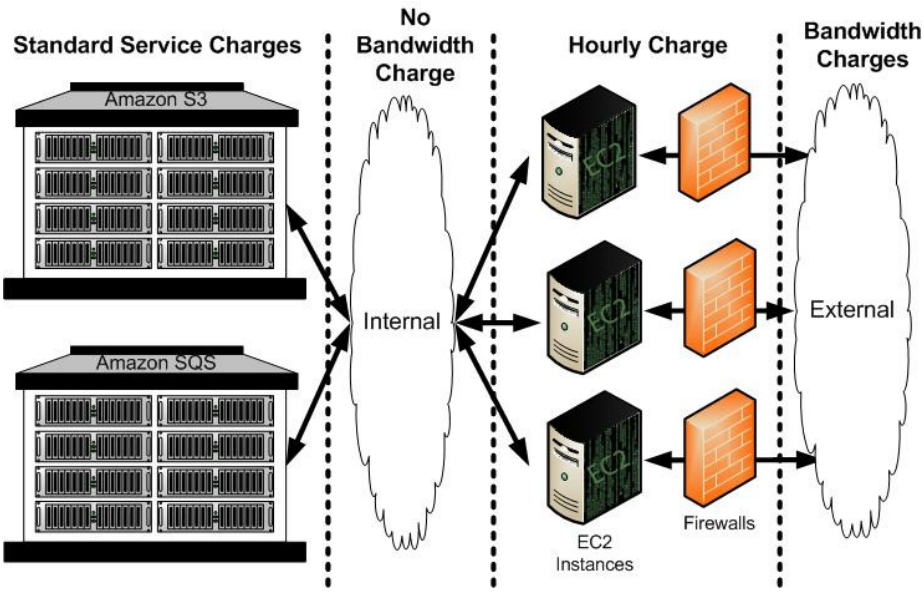


Figura 12 I costi relativi al trasferimento dei dati vengono applicati solo per le comunicazioni verso l'esterno

Costi per la trasmissione dei dati

Amazon invita tutti gli utenti di EC2 ad utilizzare tutti gli altri servizi web ad esso collegati, come S3 e SQS, infatti tutte le richieste inoltrate da un'istanza a tali servizi non viene fatturata, a differenza invece delle trasmissioni verso la rete pubblica, che invece hanno un costo calcolato in base alla quantità di traffico generata.

Regions e Availability Zones

Amazon dispone di quattro principali data center, dislocati in locazioni diverse del pianeta: due si trovano negli *Stati Uniti* (north Virginia e North California), uno in *Europa* (Irlanda) e uno in *Asia* (Singapore); queste 4 locazioni vengono definite *Regions.* È possibile decidere in quale data ceter saranno avviate le proprie istanze, questo permette di poter utilizzare quello geograficamente più vicino ai clienti, limitando la latenza e potendo rispettare alcune leggi dei paesi in cui si sta operando, ad esempio per il rispetto dei dati personali in Europa si richiede che la macchina che mantiene in memoria tali informazioni sia fisicamente all'interno della UE. Un altro vantaggio di poter scegliere la "region" delle proprie istanze è quello di poter costruire un sistema *fault-tolerant* altamente sicuro, dislocando le proprie istanze ridondate in data center diversi.

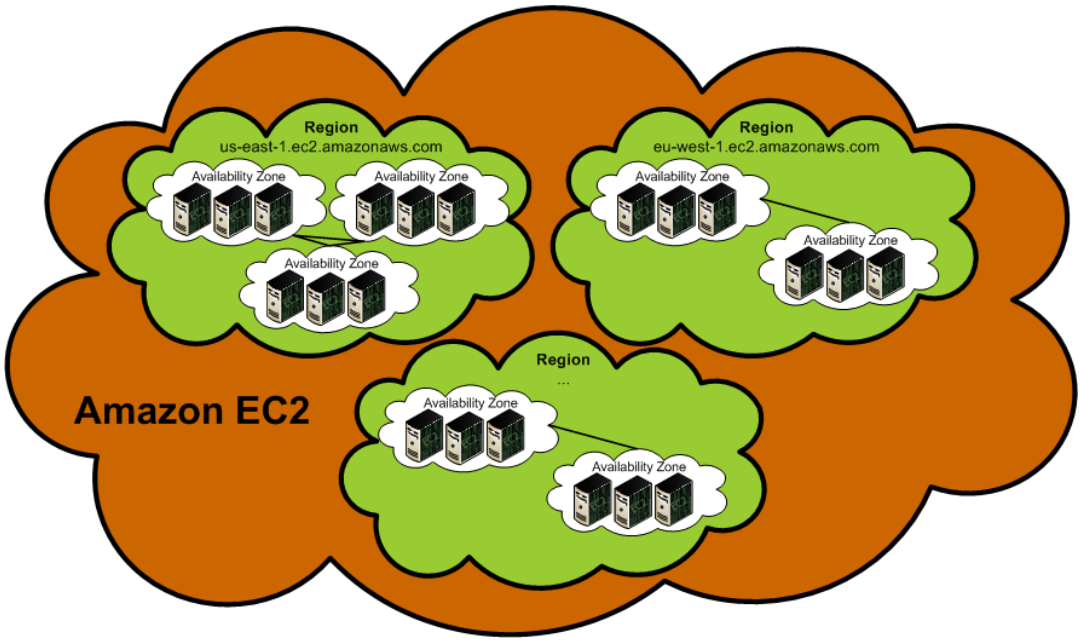


Figura 13 Distribuzione delle Regions e delle Availability Zones di Amazon EC2

All'interno di ogni Region vi sono varie *Availability Zones*, ognuna delle quali è costituita da una parte delle macchine del data center organizzate in modo da essere completamente indipendenti dalle altre, sia per quanto riguarda la fornitura dell'energia elettrica, sia per i collegamenti di rete. In questo modo è possibile isolare qualsiasi tipo di malfunzionamento all'interno di una singola zona; anche in questo caso conviene distribuire le proprie macchine in Availability Zones differenti, proteggendo le applicazioni dai malfunzionamenti che si possono verificare.

4.1.5 Chiavi di accesso – Keypair

Prima di avviare un'istanza, EC2 può modificare l'immagine originale per inserire una parte di una coppia di chiavi, che consentirà poi all'utente di accedere alla macchina virtuale tramite ssh. Ogni utenti può definire diverse *keypair*, ad ognuna delle quali si deve assegnare un nome, che deve essere passato come parametro durante la richiesta di startup di una nuova istanza. L'utilizzo di questo sistema di autenticazione permette di evitare di accedere alle macchine tramite password, metodo sconsigliato in modo da massimizzare il livello di sicurezza degli accessi; inoltre in questo modo si possono utilizzare tranquillamente immagini pubbliche, senza correre nessun rischio di intrusioni non consentite, infatti in tale tipologia di immagini l'accesso via password è disabilitato, di conseguenza si potrà accedere alla macchina solamente utilizzando una delle coppie di chiavi che l'utente stesso ha definito.

4.1.6 Interfaccia Utente

Amazon fornisce diversi strumenti con cui un utente può interagire con il sistema, tutti equivalenti tra loro: i servizi web espongono delle API, che possono essere interrogate tramite richieste web di tipo SOAP o REST da ogni utente che, fornendo le proprie credenziali, può inviare le proprie richieste al sistema. Ci sono vari strumenti che implementano tali standard, uno dei più utilizzati è *ec2ools*, usa suite di comandi utilizzabili da shell; è stato sviluppato anche un plugin per Frefox, denominato ElasticFox, che permette di utilizzare un'interfaccia grafica per interagire con il sistema. Amazon ha rilasciato anche un'interfaccia Web: *AWS Management Console*, da cui si può accedere non solo ai servizi EC2, ma anche a tutti gli altri web services presenti sul portale di Amazon, è sufficiente iscriversi gratuitamente per poter utilizzare questa interfaccia.

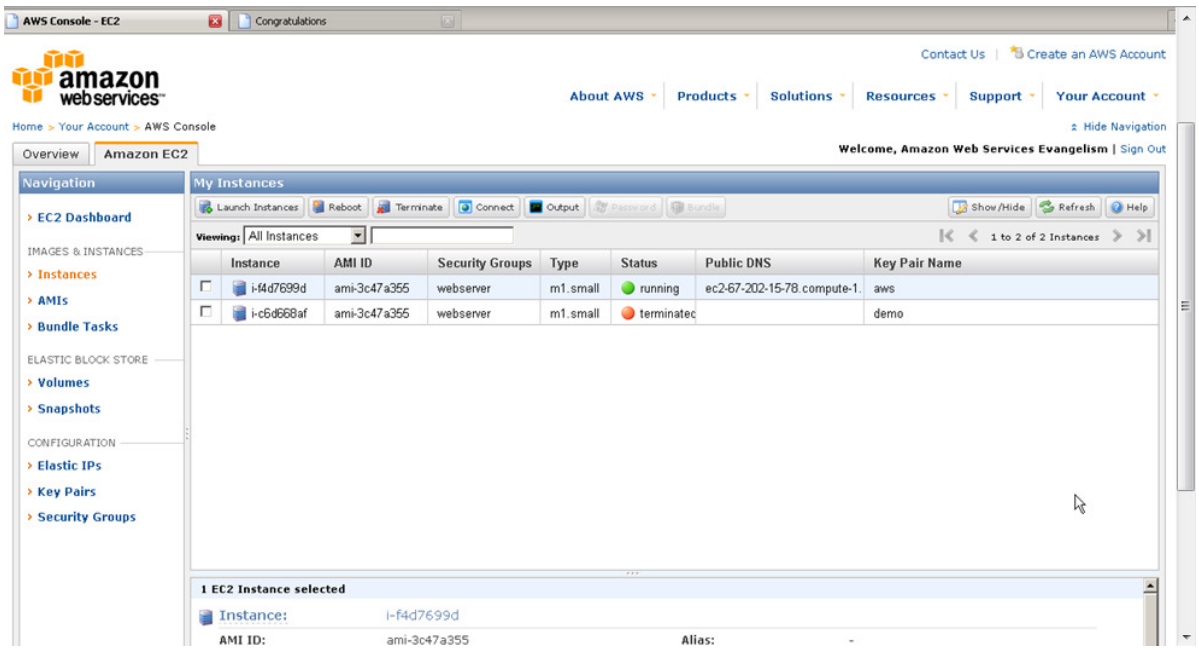


Figura 14 AWS Management Console

4.2 Esempio: avvio e spegnimento di un'istanza

Utilizzare il servizio EC2 di Amazon risulta molto semplice anche utilizzando ec2ools da linea di comando, con cui è possibile visualizzare lo stato delle proprie istanze e avviarne di nuove.

Per prima cosa si deve scegliere quale immagine utilizzare: con il comando *ec2-describe-instances* è possibile ottenere una lista delle AMI disponibili (i parametri –o self e –o amazon) indicano di mostrare sia le immagini private che quelle pubbliche

```
PROMPT> ec2-describe-images -o self -o amazon
IMAGE ami-5bae4b32 ec2-public-images/getting-started.manifest.xml amazon
available public
IMAGE ami-68ae4b01 ec2-public-images/fedora-core4-base.manifest.xml amazon
available public
IMAGE ami-69ae4b00 ec2-public-images/fedora-core4-apache-mysql.manifest.xml
amazon available public
IMAGE ami-6dae4b04 ec2-public-images/fedora-core4-apache.manifest.xml
amazon available public
IMAGE ami-6fae4b06 ec2-public-images/fedora-core4-mysql.manifest.xml amazon
available public
IMAGE ami-61a54028 <your-s3-bucket>/image.manifest.xml 495219933132
available private
```

Una volta scelta l'immagine che si vuole utilizzare, per avviare l'istanza si deve generare una nuova coppia di chiavi, se non è stata ancora definita, utilizzando il comando *ec2-add-keypair*, fornendo anche il nome da associare a questa. Il comando ritornerà il valore della chiave che dovrà poi essere utilizzata durante l'accesso alla macchina

```
PROMPT>  ec2-add-keypair gsg-keypair
KEYPAIR gsg-keypair
1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f
-----BEGIN RSA PRIVATE KEY-----
MIIEoQIBAAKCAQBuLFg5ujHrtm1jnutSuoO8Xe56LlT+HM8v/xkaa39EstM3/aFxTHgElQiJLChp
HungXQ29VTc8rc1bW0lkdi23OH5eqkMHGhvEwqa0HWASUMll4o3o/IX+0f2UcPoKCOVUR+jx71Sg
```

Una volta completate queste operazioni preliminari non resta che inviare la richiesta di avvio della macchina al sistema, fornendo il codice dell'AMI scelta e il nome della keypair da utilizzare, oltre al tipo di istanza che si vuole avviare. Il comando utilizzato è *ec2-run-instances*, che ritorna un codice identificativo dell'istanza.

```
PROMPT>  ec2-run-instances ami-5bae4b32 -k gsg-keypair
RESERVATION     r-fea54097  495219933132  default
INSTANCE        i-10a64379   ami-5bae4b32     pending   gsg-keypair  0
```

Ora si deve solo attendere che le procedure di avvio della macchina virtuale vengano completate perché questa risulti attiva, si può verificare lo stato delle proprie istanze utilizzando il comando ec2-describe-instances. Una volta che lo stato della macchina è "running" questa è attiva e vi si può accedere tramite ssh utilizzando la chiave di accesso scelta precedentemente.

```
PROMPT>  ec2-describe-instances i-10a64379
RESERVATION     r-fea54097  495219933132
INSTANCE        i-10a64379  ami-5bae4b32   ec2-72-44-33-55.z-2.compute-
1.amazonaws.com    domU-12-34-31-00-00-05.z-2.compute-1.internal
running   gsg-keypair  0
```

4.3 Soluzioni Implementative – Persistenza dei dati

La struttura di Amazon EC2 impone in molti casi di organizzare le proprie applicazioni e i propri sistemi in maniera diversa da come si farebbe se si volesse implementare lo stesso servizio su macchine reali. Le soluzioni che si possono scegliere sono diverse, ma quasi tutte richiedono una maggior complessità architetturale, soprattutto dal punto di vista della gestione dei dati delle applicazioni; tuttavia a fronte di tale aumento di complessità si riscontrano dei miglioramenti nella gestione e nel mantenimento dell'hardware e del software. Tra le principali problematiche che si devono risolvere in un sistema EC2 vi è la *persistenza dei dati*, che deve essere gestita in maniera appropriata, utilizzando i vari strumenti offerti da Amazon.

Il mantenimento della persistenza dei dati in sistemi IaaS strutturati come Amazon EC2 e Eucalyptus deve essere gestito in maniera particolare, dato che di default le istanze sono semplicemente delle copie delle immagini originarie, e tutti i dati che producono vengono persi una volta che terminano la loro esecuzione. Per salvare i propri dati, gli utenti devono utilizzare i servizi complementari di Amazon, o altri servizi analoghi se si utilizzano altri fornitori IaaS; si può utilizzare S3, che consente di mantenere i propri dati in una sorta di repository remoto, oppure utilizzare EBS, salvando i propri dati su volumi di storage che devono essere collegati alle istanze in esecuzione. La scelta di questo modello di gestione dei dati può sembrare troppo complesso ad una prima analisi, così facendo però è possibile disaccoppiare completamente la macchina dai dati che essa elabora, consentendo in questo modo di creare dei cluster largamente scalabili, gestibili in maniera più semplice e veloce. Supponendo che si voglia creare un cluster di web server, nel momento in cui si volesse aggiungere una nuova macchina affinché lavori in parallelo alle altre, è sufficiente avviare una nuova istanza EC2 con gli stessi parametri degli altri web server, senza dover clonare l'intera macchina come invece si richiede nei sistemi gestiti in maniera tradizionale. Si ha anche la possibilità di non sapere a priori di quante macchine sarà composto un cluster, infatti si può avviare la stessa immagine di una macchina più volte, senza doverne creare di nuove ogni volta.

Il modello architetturale proposto da Amazon, costituito da istanze dotate di storage volatile, consente anche di facilitare le operazioni di manutenzione e di aggiornamento dei sistemi installati, grazie alla separazione che vi è tra il software, i dati e la macchina. La separazione tra macchina e software permette di poter effettuare degli "upgrade" hardware delle istanze semplicemente riavviandole scegliendo una tipologia di istanza diversa, dotata di un maggior numero di risorse; per eseguire questa operazione infatti non è necessario modificare in alcun modo l'immagine utilizzata, le impostazioni del sistema operativo o del software installato. Avere la possibilità di effettuare rapidi aggiornamenti delle risorse allocate permette di dimensionare le proprie macchine per utilizzarle al massimo evitando

comunque fenomeni di underprovisioning, che possono essere risolti riavviando l'istanza con una quantità maggiore di risorse. Ogni utente può pagare solamente per le risorse di cui effettivamente ha bisogno e, qualora fosse necessario, ne può richiedere di aggiuntive nel tempo di un riavvio. La separazione tra software e dati permette di effettuare operazioni di manutenzione e aggiornamento sul software dell'istanza, senza doversi preoccupare della gestione dei dati e del mantenimento della disponibilità del servizio. Le operazioni di aggiornamento possono essere fatte su un'istanza avviata separatamente rispetto a quelle utilizzate per fornire il servizio, che verrà poi salvata e memorizzata nel repository di immagini; per rendere effettivi gli aggiornamenti basta anche in questo caso avviare una nuova istanza per sostituire quella obsoleta, gestendo la migrazione dei dati semplicemente spostando il volume EBS sul quale sono memorizzati da un'istanza all'altra, oppure sincronizzandoli con il bucket S3 che le contiene.

I principi di utilizzo descritti sono alla base di un servizio molto apprezzato dagli utenti: *Autoscaling*, che in automatico avvia delle nuove macchine quando verifica che il cluster in questione è sovraccarico. Il sistema non fa altro che avviare istanze aggiuntive completamente identiche alle precedenti, che andranno a contribuire alla scalabilità orizzontale del sistema. Il processo può avvenire anche nel senso opposto: Autoscaling terminerà una o più macchine del cluster, non dovendosi preoccupare di dover salvare i dati in esse contenuti. Ovviamente le istanze che fanno parte di cluster autoscalabili devono essere appositamente configurate, solitamente tramite degli script di boot, che consentano alle istanze appena avviate di unirsi al cluster e di recuperare i dati necessari per l'elaborazione (nel caso di un server web si deve contattare il load balancer e scaricare i dati, come pagine html, script php, ecc. da un repository condiviso).

Architetture per implementare la persistenza dei dati

Esistono diverse soluzioni implementative per mantenere la persistenza dei dati su istanze EC2, a partire da metodi semplici a sistemi più complessi [45]. Una soluzione molto semplice è quella di utilizzare un *volume EBS* e di salvare su di esso i dati, ad esempio montandolo al posto della cartella in cui sono contenute le tabelle di un database relazionale. Questo sistema è molto semplice da implementare, anche se risulta poco adatto nel caso in cui si volessero utilizzare più istanze della stessa applicazione, per rendere il sistema più tollerante ai guasti; infatti ogni macchina dovrà essere dotata di un volume diverso dalle altre, il che comporta un aumento della difficoltà nella sincronizzazione dei dati [46].

Un'altra tecnica che può essere utilizzata sfrutta la possibilità di creare degli *snapshot* dei volumi EBS e di poter creare dei nuovi dischi a partire da queste immagini. In questo modo si può utilizzare uno snapshot "master" dai quali si andranno a creare nuovi volumi ogni volta che una macchina verrà avviata. Questo approccio consente di separare il sistema dell'istanza dall'applicazione e dai dati che essa elabora, anche se è una soluzione poco funzionale, soprattutto perché si devono ogni volta creare

nuovi volumi a partire da uno o più snapshot. Utilizzare i volumi per la persistenza dei dati risulta comunque molto valido per applicazioni quali i database.

Si può infine utilizzare il *servizio S3* di Amazon per mantenere un repository aggiornato di tutti i dati che le nuove istanze dovranno utilizzare, basta configurarle in modo che dopo l'avvio contattino S3 per ricevere i dati necessari. Questo sistema è molto utile per la gestione di tutti i file di configurazione delle applicazioni, in quanto basta aggiornare il file su S3 per renderlo utilizzabile immediatamente. Esistono tools come *s3sync* che sono in grado di mantenere sincronizzate intere cartelle di dati tra il repository S3 e la macchina EC2. Una delle applicazioni che più si adatta a questa soluzione è costituita dai web server, che possono essere resi operativi in pochi minuti scaricando i file di configurazione e le pagine HTML dal repository S3.

In generale è necessario dotare le proprie immagini EC2 di script in boot in grado di impostare i parametri per l'esecuzione delle applicazioni, questo procedimento è facilitato dalla possibilità di passare ad ogni istanza dei dati, chiamati *user-data*, anche sotto forma di file binari, accessibili dall'interno della macchina virtuale tramite delle richieste a un indirizzo ip particolare (nel caso di Amazon EC2 è 169.254.169.254). Una tecnica molto utilizzata è di inserire nei parametri un archivio contenente gli script da eseguire, che possono essere quindi facilmente aggiornati dall'amministratore dell'applicazione.

5. EUCALYPTUS

Eucalyptus [47] è un progetto Open Source, sviluppato dall'università di Santa Barbara (California), che implementa un framework per il Cloud Computing IaaS; il sistema permette agli utenti di eseguire e controllare macchine virtuali (denominate Istanze) la cui esecuzione avviene interagendo con i diversi componenti hardware gestiti da Eucalyptus. L'architettura è organizzata in maniera modulare e gerarchica, inoltre ogni componente è stato sviluppato per essere il più semplice da installare e il meno intrusivo possibile. L'interfaccia utente è stata progettata per essere completamente compatibile con le API di Amazon Elastic Compute Cloud (EC2). Il sistema viene fornito in due versioni, una open source e una Corporate, con alcune funzionalità aggiuntive (principalmente relative a compatibilità con altre piattaforme commerciali).

5.1 Architettura del sistema

Eucalyptus è costituito da diversi elementi: è presente un componente centrale, il *Cloud Controller*, che si occupa di interagire con l'utente e conosce l'intero stato del sistema, esso controlla una serie di componenti denominati *Node Controller*, destinati all'esecuzione vera e propria delle Virtual Machines, tali nodi sono raggruppati in cluster. Sono presenti poi dei componenti per la gestione dello storage delle immagini e dei dischi virtuali: *Walrus* e *Storage Controller*. Le comunicazioni tra i diversi componenti avvengono tramite richieste a servizi Web, pubblicati da ogni elemento di Eucalyptus, che costituisce un web service autonomo; così facendo è possibile garantire buoni livelli di sicurezza per le comunicazioni tra i componenti di Eucalyptus e tra il Cloud Controller e gli utenti, sfruttando delle funzionalità già sviluppate per servizi web come *WS-Security*.

Eucalyptus non fornisce un proprio sistema per realizzare la virtualizzazione, ma è in grado di utilizzare alcuni tra i più importanti e diffusi hypervisor disponibili, in particolare KVM e Xen, supportati nella versione Opensource di Eucalyptus, e VMware ,supportato solo nella versione Corporate, sarà quindi l'amministratore di sistema a scegliere quale tra questi scegliere in base alle proprie esigenze. Uno schema generale dell'architettura del sistema è rappresentato in Figura 15, nei paragrafi successivi verrà presentato in maniera dettagliata ogni componente di Eucalyptus.

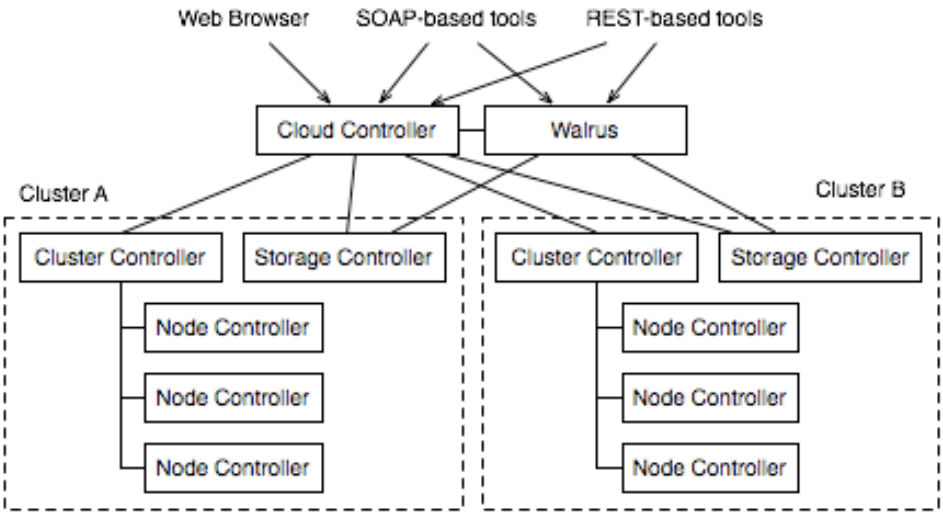


Figura 15 Architettura di Eucalyptus

5.1.1 NODE CONTROLLER(NC):

Il *None Controller* è presente su ogni macchina destinata all'esecuzione delle istanze delle VM, è necessario quindi che sia installato anche un hypervisor tra quelli supportati da Eucalyptus. La scelta di default è KVM, ma è possibile modificare un file di configurazione per scegliere le diverse alternative (Xen per la distribuzione OpenSource). Il Node Controller deve interagire con il sistema operativo installato sulla macchina, per conoscere le risorse disponibili al momento, come la quantità di memoria, il numero di cores e lo spazio disponibile su disco, vengono raccolte informazioni anche dall'hypervisor, per verificare lo stato delle istanze in esecuzione. Il componente poi fornisce queste informazioni al Cluster Controller e ha il compito di eseguire le richieste ricevute da questo, quali *runInstance* e *terminateInstance*, utilizzati rispettivamente per avviare o spegnere un'istanza. Per creare una nuova istanza, per prima cosa il Node Controller verifica l'identità del richiedente e i suoi diritti di accesso e, una volta effettuata una verifica sulle risorse disponibili, crea una copia locale delle immagini necessarie per avviare la VM (kernel, initram e il root file system) ottenendo i dati o dal componente esterno che funge da repository di immagini (Walrus) o dalla cache locale. Una volta completate queste operazioni, il NC crea un nuovo endpoint nella rete virtuale tra istanze e invoca l'hypervisor per avviare la nuova macchina virtuale.

5.1.2 CLUSTER CONTROLLER(CC)

Il *Cluster Controller* ha il compito di gestire un insieme di Node Controllers, di cui conosce la quantità di risorse disponibili e quante istanze sono attualmente in esecuzione. Le funzioni principali svolte da questo componente sono innanzitutto "schedulare" le richieste in arrivo per l'avvio di nuove istanze sui diversi NC controllati, in base alle politiche applicate, solitamente viene scelto sempre il nodo con il maggior numero di risorse libere. Un'altra funzione svolta dal Cluster Controller è quella di gestire tutte le operazioni per permettere alle macchine virtuali di comunicare tra di loro e con l'esterno: deve infatti assegnare gli indirizzi di rete appropriati e configurare appositamente le componenti. Tutte le

comunicazioni tra istanze eseguite su nodi diversi, o con la rete pubblica, devono obbligatoriamente passare dal Cluster Controller, il quale ne effettuerà il routing a seconda delle varie richieste.

Infine il *Cluster Controller* deve interagire con il Cloud Controller, ricevendo le richieste effettuate dagli utenti agendo di conseguenza, e deve fornire ad esso tutte le informazioni relative alla disponibilità di risorse del cluster, sia per quanto riguarda le risorse "fisiche" disponibili sui nodi, sia per le risorse di rete, quali gli IP pubblici disponibili.

5.1.3 WALRUS STORAGE CONTROLLER

Il componente *Walrus* realizza un servizio di data storage, la cui interfaccia è compatibile con lo standard di Amazon Simple Storage Service (S3), supporta interfacce sia REST che SOAP. In ogni sistema Eucalyptus è possibile avere un solo componente Walrus, che funge da repository di immagini, le quali vengono richieste di volta in volta dai Node Controllers secondo necessità, oltre a consentire agli utenti di salvare i propri dati. Per interagire con *Walrus*, ogni utente può utilizzare i diversi tool compatibili con S3 si Amazon, quali s3cmd o s3sync: la presenza di questo servizio non solo è fondamentale per il mantenimento in memoria delle immagini delle istanze, ma è anche utile agli utente per memorizzare tutti i dati necessari durante l'esecuzione delle proprie applicazioni.

5.1.4 STORAGE CONTROLLER

Lo *Storage Controller* fornisce un servizio compatibile con lo standard di Elastic Block Storage (EBS) di Amazon, consente di creare dei volumi (dischi virtuali) persistenti, che possono essere collegati come devices alle istanze in esecuzione. L'utilizzo di questo servizio è utile per tutte quelle applicazioni che necessitano di elaborare-salvare dati durante la loro esecuzione, è da sottolineare infatti che tutti i dati prodotti da una VM che non vengano salvati su uno di questi device non persiste dopo lo spegnimento dell'istanza. È possibile installare un solo Storage Controller per ogni Cluster.

5.1.5 CLOUD CONTROLLER

Il *Cloud Controller* è il componente principale di Eucalyptus, ha il compito di gestire tutti gli altri elementi del sistema, oltre a dover interagire direttamente con l'utente e l'amministratore tramite le diverse interfacce disponibili (WEB-based e tramite richieste a servizi web). Le sue funzioni principali sono:

- *Gestione delle risorse*: controlla lo stato generale di tutto il sistema, decidendo di volta in volta quali risorse utilizzare per ogni operazione. Consente agli utenti di scegliere le caratteristiche delle proprie istanze e della rete tra esse e controlla lo stato degli altri componenti del sistema.
- *Gestione dell'interfaccia*: fornisce i servizi per interagire con l'utente, compresi i sistemi di autenticazione e di management (tramite interfaccia web per l'amministratore).

Il Cloud Controller è l'unico il componente che conosce lo stato globale di tutta la cloud, è quindi in grado di fornire informazioni generali, quali quante VM siano in esecuzione e quante potranno ancora essere instanziate. Di seguito viene riportato lo schema della struttura di Eucalyptus:

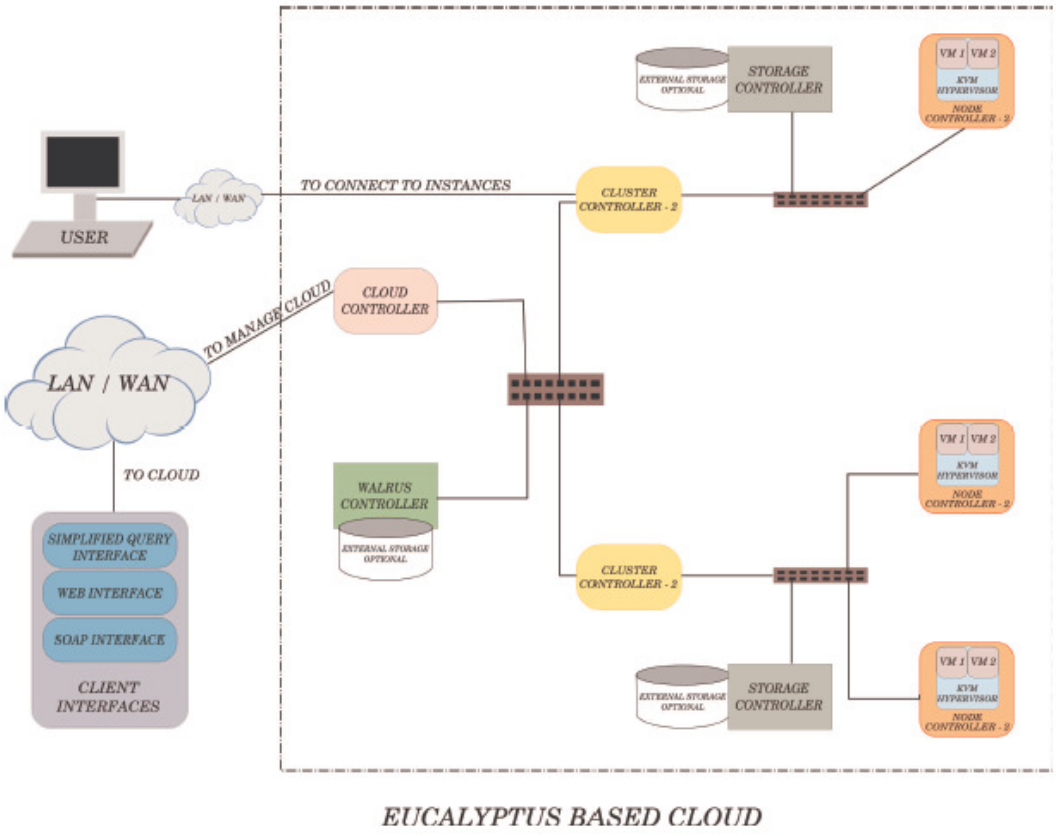


Figura 16 I componenti di Eucalyptus possono essere installati su host diversi, utilizzando la topologia rappresentata

5.2 TIPI DI VMs

Eucalyptus definisce diverse tipologie di istanze, che si differenziano in base alle risorse che verranno occupate da esse, in particolare il numero di cores su cui potranno eseguire, la quantità di memoria e lo spazio su disco ad esse dedicati. Si possono definire cinque diverse tipologie di istanze personalizzabili dall'amministratore di sistema tramite l'interfaccia web, i valori di default utilizzati sono:

Type Name	Cores	RAM	Disk Space
m1.small	1	128	2
c1.medium	1	256	5
m1.large	2	512	10
m1.xlarge	2	1024	20
c1.xlarge	4	2048	20

Tabella 10 Risorse allocate da Eucalyptus per i diversi tipi di istanze

5.3 Tipi di Networking (tra VMs)

Eucalyptus supporta diverse tipologie di interconnessione tra le istanze in esecuzione, è possibile scegliere quale di questi usare modificando i file di configurazione; non è possibile però scegliere più di una tipologia di rete contemporaneamente.

- *SYSTEM*: metodo più semplice: Eucalyptus assegna a ogni VM un MAC casuale, al quale poi verrà assegnato un indirizzo IP da un DHCP server esterno, completamente indipendente dai componenti della cloud; il DHCP server deve essere quindi appositamente impostato dall'amministratore. In questa modalità ogni macchina virtuale viene considerata come una vera e propria macchina fisica collegata direttamente alla sottorete degli altri host. Per poter funzionare correttamente in questa modalità ogni Node Controller deve avere come interfaccia di rete principale un bridge, a cui verranno collegate le istanze in esecuzione.
- *STATIC*: in questa modalità l'amministratore ha la possibilità di scegliere delle coppie statiche MAC/IP, assegnate di volta in volta alle istanze in esecuzione. Ogni VM otterrà il proprio indirizzo IP tramite un server DHCP interno, gestito direttamente dal Cluster Controller di Eucalyptus. Questo tipo di networking risulta utile se sono presenti degli indirizzi MAC/IP specifici che si vogliono assegnare alle VM.

Utilizzando SYSTEM e STATIC si perdono però alcune funzionalità tipiche di EC2, come l'isolamento del traffico tra diversi gruppi di VM, l'assegnazione da parte dell'utente di un preciso indirizzo pubblico ad un'istanza o la possibilità di modificarlo a runtime.

- *MANAGED*: è la modalità più versatile, anche se richiede alcuni vincoli nella configurazione della rete. Viene definita una rete "privata" per le VMs, irraggiungibile dall'esterno, alla quale vengono collegate tutte le istanze. È possibile inoltre definire delle "*named networks*" o "*security groups*", offrendo la possibilità di isolare gruppi di istanze da altri; nella fase di inizializzazione dell'istanze è possibile scegliere a quale security group apparterrà la macchina virtuale. Tutte le istanze appartenenti allo stesso security group saranno collegate alla stessa sottorete privata. All'interno di ognuna di queste sottoreti è possibile definire delle politiche di sicurezza utilizzando uno strumento con funzionalità simili ad iptables di linux, consentendo all'amministratore di scegliere quale tipo di traffico sia ammesso; come per la STATIC mode il sistema gestisce l'assegnazione degli IP tramite un server DHCP interno. Infine l'amministratore può mettere a disposizione una serie di "IP pubblici" che gli utenti possono associare dinamicamente alle proprie macchine (servizio simile a Amazon's elastic IPs), rendendole così accessibili anche dall'esterno.
- *MANAGED-NOVLAN*: è identico al metodo precedente, ma non consente l'isolamento tra VM, ossia la rete interna tra istanze è costituita da un unico security group.

Il *Cluster Controller* ha il ruolo di router (in MANAGED MODE) nel caso dovesse collegare VM virtualizzate su due cluster diversi, ma appartenenti alla stesso security group (ossia appartenenti alla stessa sottorete interna); tutte le comunicazioni dovranno obbligatoriamente passare da tale componente, introducendo un hop aggiuntivo nella comunicazione. Il Cluster Controller è anche il componente che implementa tutte le policy di sicurezza descritte precedentemente, che vengono applicate utilizzando l'applicativo iptables dei sistemi linux.

5.4 Creazione di Immagini

Eucalyptus supporta qualsiasi tipo di immagine compatibile con gli hypervisor utilizzati sui nodi, è possibile virtualizzare sistemi linux based o Windows (nella versione Corporate). Il formato utilizzato per le immagini è simile a quello utilizzato da Amazon, bisogna infatti fornire, oltre all'immagine della partizione di root, anche il kernel e un eventuale ramdisk. Ogni utente può scegliere se utilizzare delle immagini già predisposte per l'esecuzione su Eucalyptus, scaricabili direttamente dal sito o tramite l'interfaccia web dell'amministratore, oppure creare delle immagini personalizzate e caricarle manualmente sul sistema. È possibile convertire delle immagini di macchine virtuali esistenti, compatibili con XEN o KVM, o in alternativa creare un'immagine di una macchina attiva (reale o virtuale) utilizzando degli strumenti appositamente sviluppati come il comando *euca-bundle-vol* , del pacchetto *euca2ools* (compatibile con macchine linux). Prima di effettuare la conversione di una macchina tramite euca-bundle-vol, è necessario impostare alcuni parametri della macchina (virtuale o non) da elaborare:

- Verificare che ci sia spazio sufficiente su disco per la nuova immagine che verrà creata
- Impostare l'interfaccia di rete in modo che acquisisca in maniera automatica l'indirizzo di rete (DHCP)
- Selezionare eventuali cartelle che si vogliono escludere (parametro –e /cartella-da-escludere), molte cartelle vengono già escluse di default, quali /tmp ,/mnt, /proc
- Scelta del *kernel* e *ramdisk*: è conveniente utilizzare kernel e ramdisk compatibili con Eucalyptus, invece di utilizzare quelli relativi alla macchina da convertire, dato che questi sono spesso correlati all'hardware fisico e potrebbero provocare dei malfunzionamenti
- Moduli: alcuni sistemi richiedono che tutti i moduli presenti dell'immagine kernel compatibile siano presenti nell'immagine di cui si vuole fare il bundle, il processo più semplice per fare ciò è di montare il file system di questo kernel, ed effettuare un trasferimento dei moduli necessari
- Generare il File System table (fstab), indicante i dispositivi che devono essere montati con le rispettive opzioni. È sufficiente fornire l'opzione --generate-fstab, questa opzione è utile dato che alcuni dispositivi non hanno una corretta fstab una volta virtualizzati
- Se l'istanza da elaborare non appartiene a Eucalyptus (non è già una vm guest) o non lavora in MANAGED mode, è necessario utilizzare l'opzione –no-inherit in modo che il sistema non cerchi delle meta-informazioni caratteristiche delle immagini Eucalyptus
- Indicare la grandezza dell'immagine che si intende creare in MB

5.5 Euca2ools

Per l'utilizzo di Eucalyptus viene fornita una suite di comandi analoghi a quelli che si possono utilizzare per interagire con i servizi di Amazon EC2. Una volta installato il pacchetto *euca2ools* su di una qualsiasi macchina client sarà possibile inviare richieste al sistema Eucalyptus. I comandi sono denominati esattamente come quelli di EC2, ma con il prefisso `euca-` che sostituisce `ec2-`. Il

pacchetto richiede che vengano impostate in maniera corretta alcune variabili d'ambiente, contenenti le credenziali di accesso dell'utente al sistema Eucalyptus (chiavi, certificati, ecc).

Per un utente è possibile gestire tutte le funzionalità di Eucalyptus tramite riga di comando:

- *Gestione delle Immagini*: prima di poter eseguire un'istanza è necessario effettuare il bundle dell'immagine (viene divisa in vari pacchetti, verificata e cifrata), caricarla sul componente Walrus (S3 per Amazon) e poi registrarla sulla cloud
    - Effettuare il bundle dell'immagine si usa il comando:
      `euca-bundle-image -i nome immagine -u userid ..`
      vengono creati vari archivi contenenti l'immagine e il file manifest.xml contenente informazioni generali e checksum
    - Caricare l'immagine su Walrus:
      `euca-upload-bundle -b bucket - m image.manifest.xml`
    - Registrare un'immagine caricata:
      `euca-register imageBucket/image.manifest.xml`
    - Controllo delle VM: sono presenti alcuni comandi per visualizzare lo stato generale del sistema e delle macchine in esecuzione come
      `euca-describe-instances [-i id]`
    - Esecuzione delle VM, con il comando `euca-run-instances` fornendo key e ID del kernel del ramdisk e dell'immagine (eki eri e emi)
- Terminazione: `euca-terminate-instance -i ID`
- Riavvio: `euca-reboot-instance -i ID`
- Gestione delle autorizzazioni e delle chiavi
- Gestione delle policy di sicurezza della rete tra le VMs
- Interazione con i sistemi di Storage (EBS)

5.6 Esecuzione di un'istanza

Quando un utente richiede l'esecuzione di un'istanza, utilizzando il comando `euca-run-instances`, inviato al frontend del Cloud Controller, Eucalyptus esegue le seguenti operazioni:

- *Autentica* l'utente e verifica che abbia le credenziali adeguate per poter eseguire l'operazione richiesta (tramite verifica di chiavi precedentemente impostate)
- In base alle richieste dell'utente e alla disponibilità di risorse viene individuato il Cluster (e di conseguenza il CC) a cui verrà assegnata la macchina e il *Node Controller* su cui dovrà essere effettuata la virtualizzazione
- Viene contattato il componente *Walrus* (repository di immagini) da parte del Node Controller che effettuerà il download delle immagini necessarie per l'esecuzione dell'istanza (root, kernel e initrd)

- Viene creata l'interfaccia di rete virtuale a cui l'istanza verrà collegata, in base al tipo di networking scelto dall'amministratore e agli eventuali parametri impostati dall'utente.
- Viene lanciato il boot della Virtual Machine tramite l'hypervisor scelto per l'esecuzione (Xen o KVM per la versione Open Source di Eucalyptus)

I possibili stati in cui si può trovare una istanza sono :

- *PENDING*, quando vengono effettuate le operazioni preparatorie all'esecuzione, come autenticazione dell'utente, download e preparazione delle immagini
- *RUNNING*: la VM è in esecuzione
- *SHUTTING DOWN*: la macchina è in fase di spegnimento e si stanno effettuando le procedure per liberare le risorse occupate dalla macchina
- *TERMINATED*

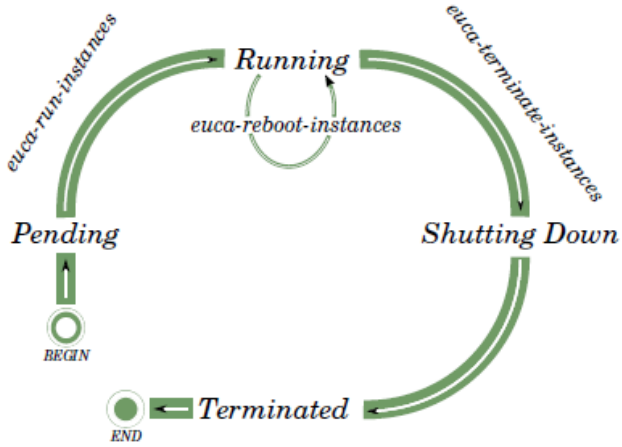


Figura 17 Ciclo di vista di un'istanza Eucalyptus

5.7 Caratteristiche e Funzionalità

Eucalyptus è un progetto ancora in fase di sviluppo, di conseguenza alcune delle funzionalità che potrebbero essere richieste da un sistema di cloud computing IaaS non sono ancora state implementate, come i servizi di Autoscaling e di load balancing. È importante sottolineare che Eucalyptus viene definito "*Hypervisor Agnostic*", infatti tutte le operazioni eseguite non considerano quale hypervisor verrà utilizzato dal Node Controller, lasciando libera scelta all'amministratore di sistema. Tuttavia l'interscambiabilità degli hypervisor non può risultare completamente trasparente all'utente, infatti le differenze strutturali tra i diversi hypervisor supportati obbligano l'utente ad utilizzare delle immagini diverse a seconda del sistema di virtualizzazione utilizzato sui NC. Ad esempio un'immagine kernel compatibile con KVM sarà molto probabilmente non compatibile con XEN (che richiede kernel appositamente modificati); Eucalyptus non è in grado di distinguere se un'immagine sia KVM o Xen-compatibile, quindi non potrà decidere autonomamente dove farla eseguire per garantirne il funzionamento. Potrebbe succedere infatti che due Node Controller appartenenti allo stesso cluster vengano impostati per utilizzare due hypervisor diversi, creando così grandi disagi all'utente: se questi

avesse immagini compatibili solo con un tipo di hypervisor, le istanze risulteranno funzionanti solamente se verranno assegnate al NC ad esse compatibile (scelta effettuata autonomamente dal Cluster Controller e non dall'utente).

Per quanto riguarda le impostazioni di *Networking*, Eucalyptus fornisce buone possibilità di personalizzazione, sia per quanto riguarda l'isolamento tra diversi gruppi di istanze e sia per l'assegnamento di indirizzi IP pubblici personalizzabili. In MANAGED MODE il sistema è in grado gestire anche la connessione alla rete esterna delle macchine, impostando tutti i parametri necessari sia sul CC che sul NC. Anche le modalità SYSTEM e STATIC possono risultare utili, visto che le istanze di Eucalyptus vengono considerate a tutti gli effetti come macchine fisiche, utilizzare tali modalità risulta semplice e immediato.

Per quanto riguarda la *scalabilità*, il sistema è progettato per gestire una quantità di host non eccessivamente elevato: si garantisce che il sistema scali bene nel caso in cui siano presenti 1 CLC, qualche CC e SC e poche decine di NC.

L'implementazione attuale di Eucalyptus non è *fault-tolerant* per nessuno dei suoi componenti, tuttavia è possibile riavviare i diversi servizi singolarmente mantenendo il sistema funzionante, è possibile anche aggiungere nuovi componenti a runtime (ad esempio aggiungere un Node Controller). Nel caso un utente volesse implementare un certo sistema di fault-tolerance per le sue applicazioni dovrà utilizzare le stesse tecniche che utilizzerebbe nel caso si utilizzasse diversi host fisici, instanziando molteplici VMs disponendole su cluster diversi. Non è presente nemmeno un sistema per garantire la *persistenza dei dati*, sia relativi alle immagini contenute nel componente Walrus , che a quelli degli Storage Controller. Entrambi questi componenti non possono essere implementati in maniera ridondante nell'architettura di Eucalyptus, ne consegue che nel caso in cui l'host su cui è installato uno dei componenti centrali (CLC o Walrus) non fosse più raggiungibile l'intera cloud risulterebbe non funzionante. Tutti i componenti di storage inoltre mantengono i dati in singola copia anche localmente.

Infine per quanto riguarda il *bilanciamento del carico*, Eucalyptus, quando riceve una richiesta per l'avvio di una nuova istanza, tende a affidare la nuova VM al Node Controller meno utilizzato al momento, tuttavia non è presente un sistema che consenta a una VM di *migrare* da un nodo all'altro. Nel caso in cui un nodo si trovasse meno impegnato di altri (nel momento in cui alcune delle sue istanze abbiano terminato la loro esecuzione), non sarà possibile bilanciare il carico del sistema utilizzandolo per eseguire alcune delle VM già in esecuzione su nodi più carichi. Non è implementato nemmeno un sistema di *load balancing* per le istanze, come Elastic Load Balancer di Amazon. Un altro servizio fornito da Amazon, ma non presente su Eucalyptus, riguarda la possibilità di definire dei cluster di istanze autoscalabili: il sistema monitora automaticamente le diverse istanze dell'utente, se queste superano un certo livello di carico, vengono avviate delle nuove macchina virtuali, garantendo così di poter gestire l'aumento del carico di lavoro da svolgere. Questo sistema funziona anche in maniera opposta, terminando l'esecuzione delle istanze quando le operazioni da eseguire diminuiscono.

5.8 Installazione

È stato installato un sistema Eucalyptus, al fine di verificarne le funzionalità e le prestazioni. La versione installata è Eucalyptus 1.6.2, su OS Ubuntu 10.04 Server.

5.8.1 Configurazione della Rete

Il sistema è costituito da un singolo Node Controller installato su una macchina VT-enabled (server2), collegata ad un'altra su cui sono stati installati gli altri componenti di Eucalyptus (server1).

Il *Server 1*, sul quale sono sono stati installati i componenti Cloud Controller, Walrus, Cluster Controller e Storage Controller di Eucalyptus, è dotato di due interfacce di rete, una collegata alla "local network" dedicata agli host eucalyptus, mentre l'altra ad una "enterprise network", con accesso Internet, attraverso la quale gli utenti possono collegarsi al CLC. Sul *Server2* è stato installato il Node controller, questo host ha una sola interfaccia di rete collegata alla "private network" e non risulta accessibile dall'esterno. Si utilizza un PC Client per accedere all'interfaccia grafica del frontend e per inviare i comandi alla cloud con il pacchetto euca2ools; non ci sono particolari requisiti per la macchina client.

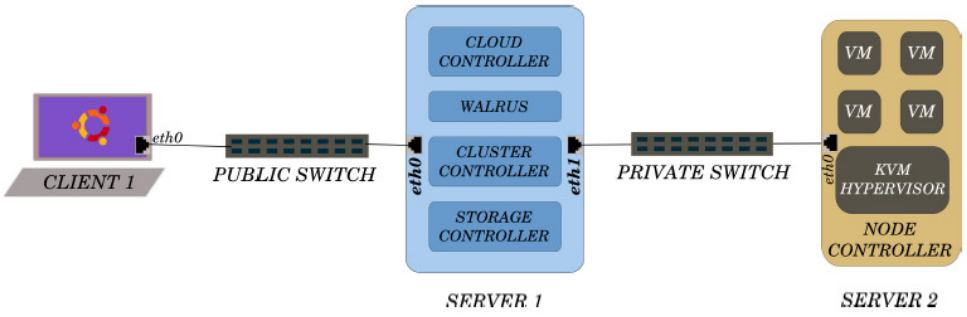


Figura 18 Architettura utilizzata per l'installazione

5.8.2 Installazione Componenti:

Per prima cosa è necessario installare una versione di Ubuntu 10.04 Server, è sufficiente mantenere le impostazioni di default seguendo il processo di installazione. Il sistema operativo deve essere dotato di alcuni servizi:

- Gli orologi dei vari hosts devono essere sincronizzati, ad esempio con ntp: `apt-get install openntpd ntpdate-debian -s` (bisognerebbe impostare il server ntp in modo che in caso di mancata connessione a internet venga utilizzato l'orologio del frontend)
- Verificare che il firewall non blocchi le porte richieste per le comunicazioni tra i componenti di eucalyptus, devono essere aperte le porte 8443, 8773, 8774, 9001 e 8775
- Per alcune Configurazioni di Rete, le macchine su cui sono presenti i Node Controllers devono avere come device di rete principale un bridge: `sudo apt-get install bridge-utils`

- Configurare correttamente l'interfaccia modificando appositamente il file */etc/network/interfaces*

Una volta installati questi pacchetti e configurate le interfacce di rete in maniera appropriata si possono installare i componenti Eucalyptus richiesti per ogni nodo, sono disponibili dei pacchetti scaricabili dai repository di ubuntu, installabili con il comando: *apt-get install eucalyptus-"nomecomponente"* (cloud-cc-sc-walrus-nc) oltre al pacchetto eucalyptus-common.

Server1

Per l'installazione dei componenti sul server1 (frontend) si procede con il comando:

```
apt-get install eucalyptus-common eucalyptus-cloud eucalyptus-cc eucalyptus-
sc eucalyptus-walrus
```

Durante l'installazione verrà chiesto di scegliere alcuni parametri:

- come impostare il sistema di mailing, possbili scelte sono:
  - Lasciare la configurazione invariata
  - Internet site: le mail verranno inviate utilizzando SMTP
  - Internet with SmartHost: le mail in entrata verranno ricevute usando smtp, quelle in uscita verranno inviata a smarthost
  - Satellite system: le mail verranno inoltrare a un altro host, che si occuperà di inviarle
- Local only: i messaggi saranno inviti solo a utenti locali
- Nome del cluster da utilizzare, ad esempio: cluster1
- Fornire una lista di IP pubblici che si possono assegnare alle VM che verranno deployate

Tutte queste impostazioni possono essere modificate una volta completata la procedura di installazione, modificando i file di configurazione appropriati.

Server2

Prima di procedere con l'installazione del Node Controller sul Server2 conviene verificare che il processore sia VT-enabled, verificando la presenza dei flag `vmx` o `svm` nel file */proc/CPUinfo*. Una volta effettuata questa verifica si può procedere all'installazione del componente tramite pacchetti:

```
apt-get install eucalyptus-nc
```

Una volta installati i componenti si deve effettuare la scelta dell'hypervisor di virtualizzazione (XEN o KVM), è sufficiente modificare i parametri presenti in */etc/eucalyptus.conf* L'hypervisor scelto per l'installazione è KVM, già automaticamente impostato da Eucalyptus

5.8.3 Registrazione Componenti

Per prima cosa si deve impostare una password per l'utente *Eucalyptus* su tutte le macchine (verrà chiesta durante la registrazione dei componenti)con il comando: `sudo passwd eucalyptus`

Dal frontend (Server1) si eseguano i comandi di registrazione:

```
/usr/sbin/euca_conf --register-walrus <front end IP address>
/usr/sbin/euca_conf --register-cluster <clustername> <front end IP address>
/usr/sbin/euca_conf --register-sc <clustername> <front end IP address>
Per registrare i nodi si esegua il comando:
/usr/sbin/euca_conf --register-nodes "<Node 0 IP address> <Node 1 IP
address> ... <Node N IP address>"
```

A questo punto si può accedere al file */etc/eucalyptus/eucalyptus.conf* e impostare la modalità di networking che si vuole utilizzare. Nel nostro caso è stata scelta MANAGED-NOVLAN

5.8.4 File di Configurazione

Tutti i parametri per configurare Eucalyptus sono contenuti in alcuni file di testo contenuti di default nella cartella /etc/eucalyptus , il file che contiene le impostazioni principali è eucalyptus.conf. Di seguito viene riportato il file utilizzato per il server1, su cui sono installati i componenti di frontend di Eucalyptus.

```
# /etc/eucalyptus/eucalyptus.conf
#
# These are the Ubuntu Enterprise Cloud's default Eucalyptus parameters.

# Affects: All
# See: **NOTE** below
EUCALYPTUS="/"
EUCA_USER="eucalyptus"
# Affects: CLC, Walrus, SC
DISABLE_DNS="Y"
DISABLE_ISCSI="Y"
JVM_MEM="512m"
# Affects: CC, NC
# See: **NOTE** below
ENABLE_WS_SECURITY="Y"
LOGLEVEL="DEBUG"
VNET_PUBINTERFACE="eth1"
VNET_PRIVINTERFACE="eth0"
VNET_MODE="MANAGED-NOVLAN"

# Affects: CC
# See: **NOTE** below
CC_PORT="8774"
SCHEDPOLICY="ROUNDROBIN"
POWER_IDLETHRESH="300"
POWER_WAKETHRESH="300"
NC_SERVICE="axis2/services/EucalyptusNC"
VNET_DHCPDAEMON="/usr/sbin/dhcpd3"
VNET_DHCPUSER="dhcpd"
#NODES="192.168.10.2"
VNET_ADDRSPERNET="32"
VNET_SUBNET="192.168.0.0"
VNET_NETMASK="255.255.255.0"
VNET_DNS="208.67.222.222"
VNET_PUBLICIPS="192.168.20.60-192.168.20.80"
# Affects: NC
NC_PORT="8775"
HYPERVISOR="kvm"
MANUAL_INSTANCES_CLEANUP=0
VNET_BRIDGE="br0"
INSTANCE_PATH="/var/lib/eucalyptus/instances/"
MAX_CORES="8"
```

Tra le impostazioni più significative si trovano:

- *MAX_CORES*: definisce quante istanze possono condividere uno stesso processore fisico
- *VNET_MODE*: definisce quale modalità di gestione della rete utilizzare
- *VNET_PUBLICIPS*: definisce il range di IP pubblici che il cluster controller può assegnare alle istanze
- *HYPERVISOR*: definisce quale hypervisor dovrà utilizzare il sistema (questa opzione si applica ovviamente solo alle macchine su cui è installato il componente Node Controller)

5.8.5 Utilizzo di euca2ools da una macchina Client.

Installare e utilizzare euca2ools per gestire il sistema via linea di comando

```
apt-get install euca2ools
```

Per utilizzare i comandi euca2ools si devono inizializzare delle variabili d'ambiente che forniscono le credenziali di accesso (certificati, key ecc) e i dati per la connessione. Per ottenere tali credenziali collegarsi via browser al sito https://CLC-ip:8443 e scaricare l'archivio zip apposito, estrarlo e impostare le variabili d'ambiente come nel file eucarc presente.

Si può verificare lo stato del sistema tramite il comando euca-describe-availability-zones:

```
euca-describe-availability-zones verbose
AVAILABILITYZONE cluster1 10.0.5.10
AVAILABILITYZONE |- vm types free / max CPU ram disk
AVAILABILITYZONE |- m1.small 0013 / 0016 1 128 2
AVAILABILITYZONE |- c1.medium 0013 / 0016 1 256 5
AVAILABILITYZONE |- m1.large 0006 / 0008 2 512 10
AVAILABILITYZONE |- m1.xlarge 0006 / 0008 2 1024 20
AVAILABILITYZONE |- c1.xlarge 0002 / 0004 4 2048 20
```

Upload di immagini

Per caricare le proprie immagini su Eucalyptus se devono seguire tre passi: *bundle, upload e register*

Esempio: ubuntu 32-bit scaricato dal sito di Eucalyptus

```
bazzu@bazzu-desktop:~/euca-ubuntu-9.04-i386$ euca-bundle-image -i kvm-
kernel/vmlinuz-2.6.28-11-server --kernel true
Checking image
Tarring image
Encrypting image
Splitting image...
Part: vmlinuz-2.6.28-11-server.part.0
Generating manifest /tmp/vmlinuz-2.6.28-11-server.manifest.xml
bazzu@bazzu-desktop:~/euca-ubuntu-9.04-i386$ euca-upload-bundle -b image-
bucket3 -m /tmp/vmlinuz-2.6.28-11-server.manifest.xml
Checking bucket: image-bucket3
Creating bucket: image-bucket3
Uploading manifest file
Uploading part: vmlinuz-2.6.28-11-server.part.0
Uploaded image as image-bucket3/vmlinuz-2.6.28-11-server.manifest.xml
bazzu@bazzu-desktop:~/euca-ubuntu-9.04-i386$ euca-register  image-
bucket3/vmlinuz-2.6.28-11-server.manifest.xml
IMAGE eki-C7BD1486
```

Operazioni Analoghe devono essere effettuate per il ramdisk e per il root file system

Avvio di Istanze

Per avviare un'istanza è sufficiente utilizzare il comando *euca-run-instances* , fornendo i parametri necessari, quali l'ID delle immagini da avviare e le credenziali di accesso

```
euca-run-instances -k testkey --kernel eki-34323333 --ramdisk eri-33344234
emi-53444344
```

Si può monitorare lo stato delle proprie istanze utilizzando euca-describe instances:

```
euca-describe-instances
RESERVATION r-3494070E admin default
INSTANCE i-3E33080E emi-3F800D65 192.168.0.220 172.19.2.2 running 0
c1.medium 2010-04-21T06:09:53.306Z cluster1 eki-31610D46 eri-BEBD1027
```

In questo caso la macchina virtual sarà accessibile dall'esterno all'indirizzo 192.168.0.220

Utilizzo di Storage Devices (EBS)

Per collegare ad un'istanza un disco virtuale si utilizza il comando *euca-attach-volume* fornendo l'id dell'istanza e il nome del device a cui sarà associato.

```
euca-attach-volume -i i-99838888 -d /dev/sdb vol-33534456
```

Ora dalla VM sarà sufficiente montare il disco per renderlo accessibile in lettura e scrittura.

Per scollegare un device si utilizza *euca-detach-volume*

5.8.6 Utilizzo del Server DNS

Eucalyptus è dotato di un *server DNS interno*, in modo da poter assegnare un nome ad ogni istanza. Per attivarlo si deve modificare appositamente il valore di *DISABLE_DNS* in *DISABLE_DNS='N'* e quindi riavviare il servizio eucalyptus-cloud. Tramite l'interfaccia web di amministrazione (https://<ip clc>:8443) è possibile scegliere quale sottodominio dovrà essere utilizzato da Eucalyptus; le istanze verranno mappate come `euca-A.B.C.D.eucalyptus.<subdomain>`

5.9 Inizializzazione di un'istanza e inserimento delle chiavi

Ogni utente, per poter poi accedere alle proprie istanze deve creare almeno una coppia di chiavi, tramite i comandi *euca-add-keypair* e *euca-delete-keypair*; tale coppia di chiavi deve essere fornita per collegarsi alle istanze tramite ssh. Quando una macchina viene creata, viene indicato anche il nome della keypair da associare ad essa. Prima di avviare un'istanza il sistema inserisce alcuni file nell'immagine del disco che verrà utilizzato dalla VM; in questo modo sarà poi possibile accedere all'istanza dall'esterno, tramite ssh, utilizzando la chiave corrispondente a quella fornita in fase di inizializzazione della macchina. L'operazione avviene prima di avviare la virtual machine, Eucalyptus monta il file immagine del disco in una cartella temporanea e inserisce il file */root/.ssh/authorized_keys* , mediante uno script perl */urs/share/eucalyptus/add_key.pl*

5.10 Personalizzazione delle immagini

Per creare delle immagini *personalizzate* da utilizzare su Eucalyptus, o su EC2, si possono utilizzare diversi metodi, di cui il più semplice e funzionale consiste nel modificare le immagini di base fornite dal provider. Il procedimento consiste nel procurarsi i file iso di "base", che di solito contengono un'installazione del sistema operativo minimale, ottimizzati per essere eseguiti sulla piattaforma IaaS; successivamente si monta tale immagine per poterla utilizzare come root su un PC locale. A questo punto è possibile effettuare tutte le modifiche necessarie che si vogliono rendere permanenti nel file immagine preso in considerazione; una volta completate tutte le operazioni desiderate basta caricare il file immagine modificato su Eucalyptus per rendere operativa la nuova immagine modificata. Di seguito verranno descritte le operazioni da eseguire in maniera dettagliata:

- Per prima cosa si deve collegare l'immagine a un device di *loopback*:
  `sudo losetup /dev/loop5 ubuntu.9-04.x86-64.img`
- Si deve poi *montare* l'immagine in una cartella temporanea:
  `sudo mount /dev/loop5 tmp-mnt/`
- Lanciare il comando `chroot`, per modificare la *root directory* del sistema e utilizzare quella dell'immagine da modificare `sudo chroot tmp-mnt/` Questo comando funziona solamente se l'architettura dell'host utilizzato è compatibile con quella richiesta dall'immagine che si vuole modificare, di conseguenza non sarà possibile modificare immagini di sistemi operativi a 64 bit utilizzando host a 32 bit.
- Una volta effettuate le operazioni appena descritte si possono eseguire tutte le *modifiche* richieste sull'immagine, eseguendo i comandi come se questi venissero lanciati sulla macchina virtuale; è possibile quindi installare pacchetti, aggiornamenti software, ecc. Quando tutte le operazioni sono state completate si può reimpostare la root del sistema semplicemente lanciando il comando exit
- Le ultime operazioni da eseguire richiedono di *smontare* l'immagine, al fine di evitare problemi di inconsistenza dei dati*:*
  `sudo umount -l tmp-mnt/`
  `sudo losetup -d /dev/loop5`
- L'immagine iso è ora modificata, si deve procedere a effettuare il *bundle* e l'*upload* sulla piattaforma IaaS.

5.11 Problemi

Durante l'installazione e la configurazione del sistema, sono emerse alcune problematiche ricorrenti, elencate di seguito con relative soluzioni:

- Può succedere che Eucalyptus non veda alcuno slot disponibile per le istanze: il comando `euca-describe-availability-zones` verbose ritorna un numero massimo di macchine pari a 0. Per risolvere il problema è sufficiente riavviare il servizio *eucalyptus-cc* sui cluster

controller interessati (probabilmente in fase di boot il cluster controller non è riuscito a comunicare con il node controller)

- Risulta impossibile lanciare le istanze, il node controller mostra l'errore: `libvirt: internal error no > supported architecture for os type 'hvm' (code=1)` Si devono riavviare i servizi *libvirt* e *eucalyptus-nc*, in fase di avvio Eucalyptus ha tentato di collegarsi a libvirt, quanto questi non era ancora pronto.
- Quando ci si collega ad un'istanza tramite ssh, si riceve l'errore:

  ```
  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
  @ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @
  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
  IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
  […]
  Permission denied (publickey,password).
  ```

  Questo errore si presenta perché le istanze cambiano il loro IP ogni volta che vengono avviate, è necessario disabilitare questo tipo di controllo di sicurezza su ssh del client, accedendo al file: `/etc/ssh/ssh_config` e modificando il valore `StrictHostKeyChecking=no`
- Errore in fase di collegamento di un EBS (volume) ad un'istanza, con messaggio: `[EUCAERROR] libvirt: operation failed: adding scsi disk failed (code = 9)` Bisogna controllare che il network device associato allo storage controller sia quello collegato alla rete privata della cloud (quella in cui sono presenti i node controllers)

6. ALTRI SOFTWARE IAAS

6.1 Eucalyptus Enterprise

È disponibile anche una versione commerciale di Eucalyptus, basata sul prodotto opensource, ma con alcune caratteristiche aggiuntive, che consentono di implementare piattaforme IaaS private compatibili con i più diffusi standard commerciali. Oltre ad avere tutte le funzionalità del prodotto opensource, *Eucalyptus Enterprise Edition* [48] può interfacciarsi con altri sistemi commerciali, fornisce sistemi di accounting e gestione delle tariffe, inoltre garantisce migliori prestazioni di I/O con il supporto per lo Storage Area Network.

- Tra gli hypervisor utilizzabili si aggiunge il supporto a VMware e alla sua piattaforma vSphere. I clienti potranno quindi sfruttare tutte le diverse tecnologie di virtualizzazione presenti nei loro data center, coordinandole tramite una singola piattaforma gestita da Eucalyptus in maniera completamente trasparente all'utente.
- È possibile eseguire sistemi guest sia Linux che Windows (incluso il servizio di remote desktop), non supportato nella versione Opensource.
- L'integrazione con i sistemi di storage SAN, utilizzabile sia per i componenti Walrus (S3) che per lo Storage dei volumi EBS, consente di ottenere prestazioni di I/O migliori, fondamentali per i tempi di risposta del sistema.
- È presente un sistema di gestione di gruppi e account utente molto personalizzabile, consentendo il controllo delle risorse di calcolo, storage e di rete assegnate a ciascun utente. Associato a questo servizio è presente un gestore di quote e di rendiconto, che conteggia le risorse assegnate da ciascun cluster agli utenti; questo componente fornisce anche strumenti di analisi generale dello stato del sistema.

Si può ottenere, previo richiesta, una versione trial del software per un periodo di test; i prezzi di acquisto non sono pubblicati sul sito internet, è necessario contattare l'ufficio vendite.

6.2 OpenStack

OpenStack [49] è un progetto open source, nato nell'estate 2010, con l'obiettivo di *produrre una piattaforma di Cloud Computing che possa soddisfare le esigenze delle cloud private e pubbliche indipendentemente dalla loro grandezza, che sia semplice da implementare e fortemente scalabile.* Il progetto può contare su due collaborazioni molto importanti: RackSpace (piattaforma di hosting Cloud) e la NASA; essi hanno rilasciato parte del loro software ad OpenStack, RackSpace ha contribuito con una versione open del software di gestione di Cloud Files, un servizio di storage, mente la NASA con la sua piattaforma di IaaS Nebula Cloud. Il progetto infatti si divide in due parti: *Object Storage* e *Compute*, in linea con quanto descritto sopra; entrambi questi software devono ancora essere rilasciati ufficialmente, si prevede una release a ottobre 2010 per Compute e a metà settembre per Object Storage. Le funzionalità che sono già implementate in OpenStack, grazie alle risorse fornite dai collaboratori sono:

- Gestione delle Virtual Machines
- Configurazione della rete interna
- Un sistema di storage distribuito e replicato
- Servizio di storage e accesso a dati fino a 5GB di grandezza.

Oltre a queste funzionalità, verranno aggiunte le OpenStack RESTful API per gestire l'interazione con l'utente; queste sono un'evoluzione delle API utilizzate da RackSpace, alle quali aggiungono nuove funzionalità. Per quanto riguarda l'hypervisor da utilizzare per la virtualizzazione, OpenStack ha scelto di lasciare libera scelta all'utente: saranno supportati XenServer, KVM e UML. Si potranno utilizzare anche immagini Virtualbox, consentendo agli utenti di testare in locale le loro VM prima di instanziarle sulla cloud. Il sistema sarà integrato con Object Storage, seguendo il modello già utilizzato dai diversi provider di IaaS, garantendo la possibilità agli utenti si salvare i dati elaborati dalle VM. La gestione della rete consentirà di assegnare ip pubblici alle VM e di personalizzare le politiche di sicurezza della rete interna. Al fine di facilitare gli sviluppatori a contribuire nel progetto, OpenStack mantiene le versioni del proprio codice sorgente molto documentate, vengono anche definite delle linee guarda da seguire per lo sviluppo di eventuali nuovi componenti.

6.3 Open Nebula

Open Nebula [50] è un software opensource che consente di costruire tutti i tipi di cloud IaaS: Privato, Ibrido e Pubblico; il software gestisce lo storage dei dati, la virtualizzazione e il networking tra le VM, interagendo con l'utente tramite delle API. I tre componenti principali di OpenNebula sono:

- L'*Hypervisor*, che si occupa della virtualizzazione di ogni VM; sono supportati KVM, Xen e VMWare
- *Virtual Infrastructure Manager,* per la gestione delle immagini, dell'interazione con l'utente, la gestione della rete virtuale tra istanze e della fault tolerance

- *Scheduler*, per distribuire uniformemente le risorse, facendo eseguire le VM sugli host piu adatti, assicurando i livelli previsti di SLA

La struttura del sistema è costituita da un nodo che avrà il ruolo di frontend, il quale può essere collegato a diversi altri host, che si dedicheranno alla virtualizzazione. L'interazione con l'utente avviate mediante linea di comando, utilizzando delle chiamate a RESTful API, sviluppate in linea con lo standard OCCI (Open Cloud Computing Interface) [51]. Ad oggi non è presente un'interfaccia grafica per una gestione più user friendly.

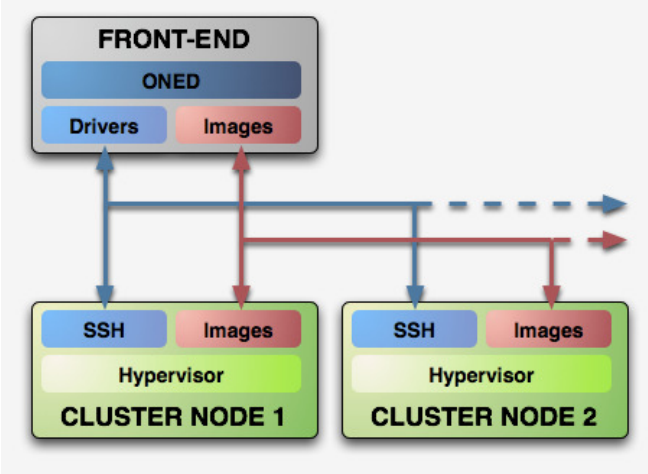


Figura 19 Architettura del sistema OpenNebula

È possibile anche creare una cloud *ibrida*, sfruttando le risorse dei propri nodi e quelle di provider esterni come Amazon EC2 o Elastic Hosting. A livello utente questo tipo di integrazione risulta trasparente, sarà infatti il sistema che deciderà autonomamente se affidare una VM a un nodo interno o a provider esterni. Per implementare questa funzionalità si deve configurare appositamente un "Adaptor" per il provider desiderato, indicando le proprie credenziali di accesso e altri parametri, quali il numero massimo di VM instanziabili.

OpenNebula risulta essere uno dei progetti open source più avanzati, visto che possiede molte funzionalità che altri software analoghi non hanno ancora. In particolare sono pienamente supportati i tre hypervisor più diffusi (Xen, KVM e VMware), con la possibilità di sfruttarne appieno il potenziale. Si possono creare nuove immagini a partire da VM in esecuzione, gestire operazioni fondamentali quali migrazione (live e non) e sospensione delle istanze. La gestione della rete virtuale è anch'essa molto personalizzabile, ogni VM può essere connessa alla rete pubblica tramite un indirizzo riservato, mentre la rete interna può essere divisa in varie zone per consentire l'isolamento tra le istanze in esecuzione. Sono presenti anche servizi per la gestione e lo sviluppo di sistemi multi tier costituiti da più istanze, in grado di auto-organizzarsi in fase di avvio: si può configurare un insieme di macchine virtuali, ognuna con un compito specifico e coordinarle come fossero un unico sistema, al fine di facilitare tutte le operazioni di gestione. La gestione della persistenza dei dati prodotti da una VM avviene in base alle preferenze dell'utente: una macchina può eseguire una copia dell'immagine originale, perdendo i dati una volta terminata la sua esecuzione, oppure salvare una nuova immagine diversa dalla precedente; in

alternativa l'hypervisor utilizzerà l'immagine originale per la virtualizzazione, mantenendo così tutti i dati in memoria una volta che la VM verrà spenta. Non è presente invece un sistema per la gestione di dischi simile a EBS di Amazon, che permetta di utilizzare un disco su una VM, aggiungendolo o togliendolo senza dover riavviare la macchina.

Reservoir

OpenNebula fa parte del progetto europeo *Reservoir* [52], che ne ha adottato il software come base di partenza per lo sviluppo di ulteriori servizi cloud. L'obiettivo del progetto è quello di sviluppare un'architettura per fornire delle piattaforme orientate ai servizi flessibili e scalabili; *Reservoir* fornirà inoltre anche un'implementazione di tale architettura. I componenti che costituiscono il sistema sono molteplici, ma la base per la gestione della virtualizzazione è stata affidata a OpenNebula, favorendone sicuramente lo sviluppo nel prossimo futuro; è stato realizzato anche un software in grado di fornire un'interfaccia grafica generale per la gestione del cloud, denominato Claudia. Questa applicazione può risultare estremamente utile per interagire con il sistema OpenNebula, che ancora non ha sviluppato nessun tipo di GUI.

6.4 Intalio

Intalio [53] ha sviluppato software in grado di fornire i più svariati servizi di cloud computing, a partire da quelli SaaS (con applicativi quali CRM e Office Suite), ai PaaS e IaaS. Utilizzando questi software ognuno ha la possibilità di creare il proprio servizio cloud che preferisce.

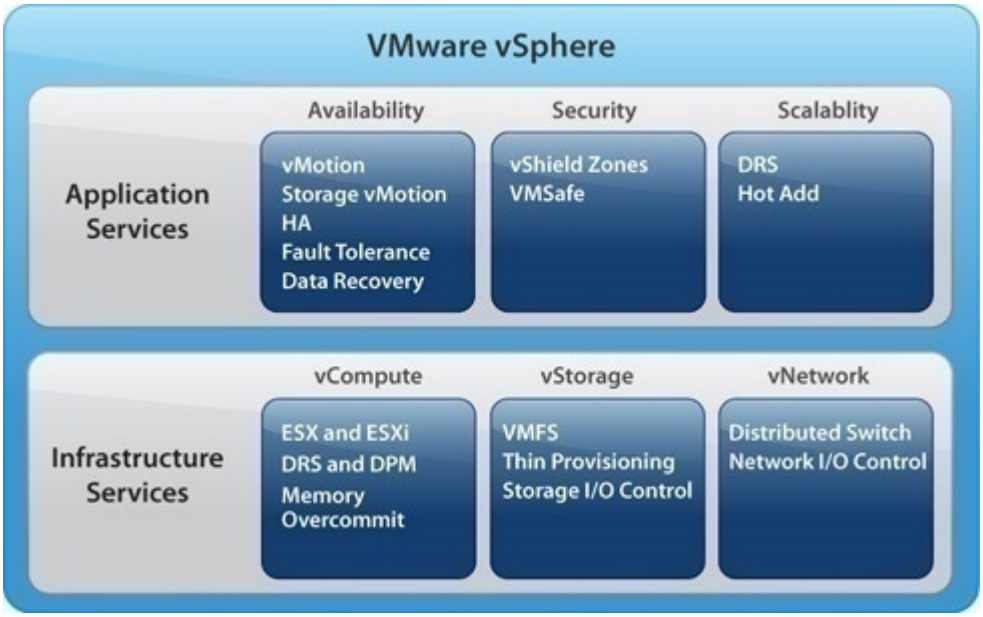
- *Elastic Database*: fornisce un database SQL remoto, ottimizzato per l'utilizzo da parte di applicazioni web. Il database verrà eseguito su una o più macchine collegate tra loro, in base al carico di lavoro richiesto in ogni particolare periodo, il tutto viene gestito autonomamente dal provider.
- *Elastic Storage*: è un servizio analogo ad Amazon S3, capace però di mantenere in memoria files più grandi di 5GB. I dati vengono automaticamente replicati su diverse macchine per garantire i livelli di fault tolerance adeguati.
- *Elastic Compute*: fornisce capacità di calcolo (Virtual Machine) on-demand, in maniera totalmente analoga ad Amazon EC2. Il supporto alla virtualizzazione viene fornito da molteplici hypervisor: VMware, Hyper-V, Xen e Virtualbox
- *Cloud Controller*: è in grado di gestire tutte le applicazioni cloud instanziate dall'utente; fornisce anche strumenti di monitoring e di fatturazione.
- *Cloud Appliance*: è un servizio per costruire una cloud privata per aziende, implementata su hardware di Intalio, tra cui si trovano dispositivi delle maggiori case produttrici: Cisco, Dell, HP e IBM

Tutti gli applicativi di Cloud sono stati sviluppati abbastanza recentemente e, nonostante siano già in produzione in alcune aziende partner di Intalio, questi non sono ancora stati rilasciati ufficialmente, il lancio avverrà durante l'inverno 2010.

6.5 vShpere (VMware)

VMware propone una soluzione commerciale per la realizzazione di cloud private basata sul suo famoso hypervisor e organizzata in modo da sfruttare al massimo i diversi tool già sviluppati dall'azienda leader nel settore della virtualizzazione. La soluzione proposta è costituita da *vSphere* [54], una piattaforma per la gestione della virtualizzazione sviluppata da VMware, considerata tra le più robuste e performanti del settore. Con vSphere è possibile:

- *Virtualizzare applicazioni* cruciali per il business di un'azienda, ottenendo affidabilità e flessibilità, garantendo la disponibilità del servizio e brevi tempi di risposta
- Creare dei *gruppi di risorse (pool)*, in modo da fornire alti livelli di Service Level Agreement (SLA), minimizzando i costi per le applicazioni.
- Mantenere un *alto livello di sicurezza*, controllo e compatibilità delle applicazioni e dei dati presenti nella cloud



- **Figura 20 vSphere fornisce diversi servizi, divisi in servizi Applizativi e Infrastrutturali**

In sistema di private cloud richiede che si garantisca un buon livello di protezione; per ottenere questo vSphere interagisce con un altro prodotto, *vShield*, una suite di applicazioni destinate alla sicurezza dei dati e delle applicazioni nella cloud. vSphere 4 è un vero e proprio sistema operativo, da installare su tutti gli host che faranno parte della cloud, comprende due tipi di servizi: i servizi di infrastrutturali e i servizi applicativi [55].

Servizi Infrastrutturali

Sono costituiti da tutti i componenti che consentono la virtualizzazione completa di macchine fisiche, storage e risorse di rete, oltre alla gestione di questi in base alle richieste e alle priorità scelte. Per la gestione della virtualizzazione efficiente delle risorse e della loro aggregazione si utilizza *vCompute*, il quale comprende diversi applicativi: VMware ESX e VMware ESXi, che si occupano della virtualizzazione; VMware Distributed Resource Scheduler (DRS) che aggrega le risorse di elaborazione dei vari cluster e le alloca in modo dinamico alle macchine virtuali in base alle priorità aziendali; VMware Distributed Power Management (DPM) che automatizza la gestione energetica dei cluster, in modo da minimizzare i consumi.

Della gestione dello storage si occupa *vStorage*, un insieme di tool atti a astrarre le risorse di storage dell'hardware sottostante e di renderlo disponibile alle macchine virtuali. Sono compresi in questo componente: VMware vStorage Virtual Machine File System (VMFS) che fornisce un cluster file system ad alte prestazioni, ottimizzando la condivisione e l'accesso simultaneo alle risorse; VMware vStorage Thin Provisioning che fornisce l'allocazione dinamica della capacità di storage, consentendo di ridurre significativamente lo spazio occupato.

Per l'organizzazione della rete interna virtuale si utilizza *vNetwork*, che comprende VMware vNetwork Distributed Switch, il quale semplifica e potenzia, l'amministrazione e il controllo di macchine virtuali in rete, fornendo agli amministratori di rete interfacce familiari per controllare la qualità del servizio a livello di macchina virtuale.

Servizi Applicativi

Forniscono controlli e funzioni necessarie per garantire la disponibilità dei dati, la sicurezza delle comunicazioni e la scalabilità del sistema. I servizi che garantiscono la disponibilità consentono di fornire applicazioni con diversi livelli di affidabilità e ridondanza, in base alle specifiche esigenze, senza dover interagire con l'hardware sottostante. Questi servizi risultano molto utili nel caso in cui si dovessero effettuare delle operazioni di manutenzione hardware o software su determinati host, in quanto consentono di spostare il carico di lavoro ad essi assegnato su altre macchine. Gli strumenti che appartengono a questa sezione sono: *VMware vMotion* che consente di migrare in tempo reale le macchine virtuali tra i server senza causare interruzioni per gli utenti o perdite di servizi; *VMware Storage vMotion* in grado di migrare in tempo reale i dischi delle macchine virtuali senza causare interruzioni per gli utenti o perdite di servizi; *VMware High Availability (HA)* che consente di riavviare in pochi minuti, in modo automatico, tutte le applicazioni immobilizzate da guasti hardware o del sistema operativo; *VMware Fault Tolerance* che assicura la disponibilità continua di tutte le applicazioni, senza causare downtime o perdite di dati in caso di guasti hardware; *VMware Data Recovery* che fornisce attività di backup e ripristino semplici e efficienti. Durante la fase di creazione delle VM, si potrà scegliere se associare alla nuova VM un disco virtuale già esistente, crearne uno

nuovo, o utilizzare direttamente una SAN, le eventuali immagini create vengono salvate in uno dei datastore disponibili, in accordo con la scelta dell'utente.

I servizi di sicurezza consentono di fornire applicazioni con un livello di sicurezza conforme alle policy in vigore beneficiando di efficienza dal punto di vista operativo, per eseguire queste operazioni si utilizzano: *VMware vShield Zones* che estende le policy di sicurezza applicativa all'ambiente condiviso mantenendo inalterata la segmentazione di utenti e dati sensibili a livello di aree affidabili e rete; *VMware VMsafe* consente di utilizzare prodotti per la sicurezza che funzionano in combinazione con lo strato di virtualizzazione per fornire alle macchine virtuali livelli di protezione più elevati rispetto ai server fisici (ad esempio è possibile installare un antivirus direttamente in vSphere, invece di collocarlo sulle VM).

I servizi per la scalabilità consentono di fornire a ogni applicazione un'adeguata quantità di risorse in base alle esigenze, senza causare interruzioni delle attività. I componenti utilizzati sono: VMware DRS che bilancia in modo dinamico le risorse allocate per i vari server; la capacità "hot add" consente di aggiungere CPU e memoria alle macchine virtuali in caso di necessità; la capacità "hot plug" consente di inserire o rimuovere dispositivi di rete e storage virtuale sulle VM; la capacità "hot extend" dei dischi virtuali consente di aggiungere storage virtuale alle VM in esecuzione. VMware vSphere include il supporto per *vApp*, un'entità logica costituita da una o più macchine virtuali, che utilizza lo standard di settore Open Virtualization Format per specificare e incapsulare tutti i componenti di un'applicazione multi-tier, unitamente alle policy operative e ai livelli di servizio associati. Questo strumento è un contenitore, che può comprendere una o più VM, ogni vApp può essere eseguita, spenta o clonata come una singola VM; si fornisce ai responsabili delle applicazioni una modalità univoca per definire i criteri di esecuzione di un'applicazione che il sistema operativo cloud è in grado di interpretare ed eseguire automaticamente, con questo sistema è possibile trasferire applicazioni nuove ed esistenti tra cloud interni o esterni basati su VMware vSphere senza modificare i livelli di servizio.

6.5.1 vSphere Client

Tutte le operazioni di amministrazione della cloud privata si effettuano tramite vSphere Client, uno strumento dotato di interfaccia grafica semplice e intuitiva. Mediante questa applicazione, l'amministratore di sistema potrà accedere a tutte le funzioni sopra descritte, sia per quanto riguarda la gestione hardware delle macchine host, che per il controllo delle VM guest. L'interfaccia è costituita da una pagina principale, contenente informazioni generali, dalla quale si accede ad altre dedicate all'utilizzo e al monitoraggio di funzioni quali: gestione delle macchine host, creazione-avvio-spegnimento di VM, High Availability, live migration, Backup, ecc.

In pochi minuti è possibile rendere l'intero sistema funzionante, anche grazie alle procedure guidate per l'impostazione dei diversi servizi, come la creazione di nuove VM o il backup periodico del loro stato di esecuzione. È da sottolineare come la maggior parte delle operazioni che un amministratore può eseguire vengano applicate a caldo, rendendo quindi il tutto completamente trasparente all'utente finale.

Con i sistemi di HA e fault tolerance si garantisce la persistenza del servizio anche in caso di guasti hardware o blocchi al sistema operativo delle VM, mediante la replicazione di macchine su host diversi (VMware fault tolerance) o il restart automatico (VMware HA); con il sistema di live migration è possibile spostare l'esecuzione delle macchine virtuali da un host ad un altro, servizio utile nel caso in cui si debba spegnere una macchina host per effettuare interventi di manutenzione. Il sistema consente inoltre di impostare backup periodici delle VM, fornendo dei punti di ripristino da utilizzare in caso di necessità.

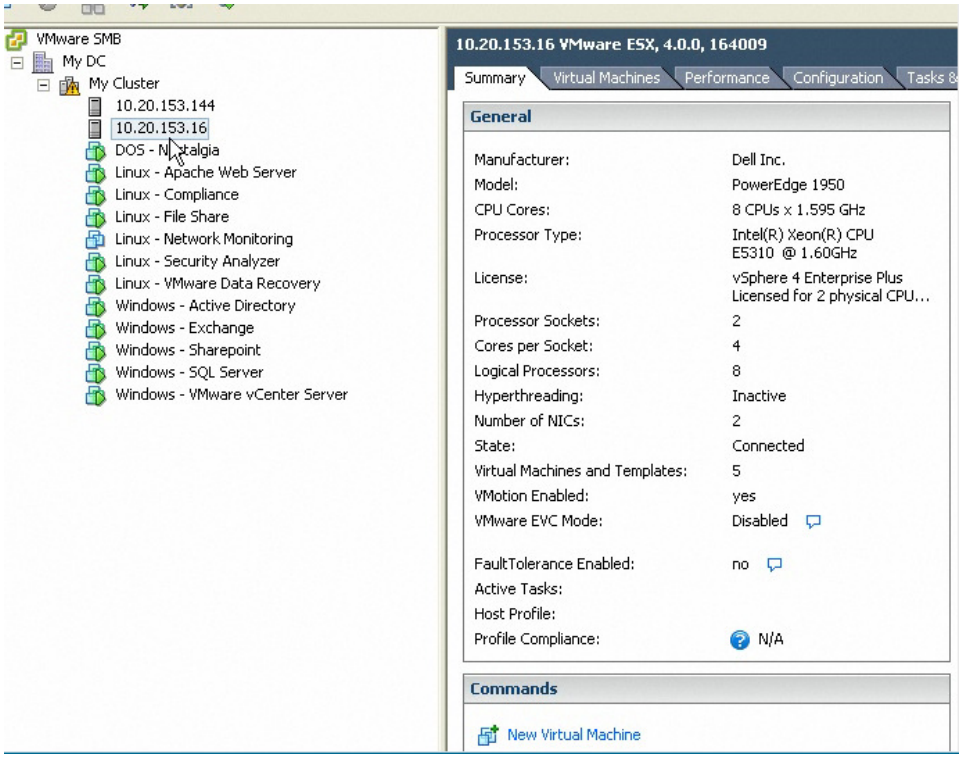


Figura 21 Pannello di controllo del cluster di host (icone grigie) e delle VM (icone verdi)

6.5.2 Versioni Commerciali di vSphere

VMware commercializza vSphere con diverse modalità di licenza, di prezzo e di funzionalità abilitate, in modo da poter soddisfare le esigenze sia delle piccole e medie imprese, che delle grandi aziende. Sono presenti infatti 4 versioni commerciali: *Essentials, Standard, Advanced Enterprise ed Enterprise Plus*; è possibile tuttavia ottenere delle licenze personalizzate se necessario.

Le licenze vSphere si differenziano innanzitutto sul numero di processori utilizzabili su ogni host fisico, la versione Standard può gestire fino a 6 core per ogni processore, per le altre questo numero aumenta fino a 12 nella la versione Enterprise. Per quanto riguarda la memoria, ogni versione può gestire fino a 256GB di RAM, tranne la versione Enterprise Plus, che non ha questo limite. Le diverse versioni (kit) di vSphere sono state pensate per rispondere alle esigenze dei diversi scenari implementativi e le diverse esigenze dei clienti:

- *vSphere Essentials Kit* – È una suite che consente di utilizzare fino ad un massimo di 3 server fisici, è dedicato a tutte quelle piccole realtà che richiedono un sistema già pronto per l'uso.

Sono compresi in questa versione tutti i componenti principali di vSphere, che consentono la gestione della virtualizzazione e dello storage.

- *vSphere Essentials Plus* – Alla suite precedente è possibile aggiungere i componenti dedicati alla foault tolerance, al backup dei dati e alla live migration, questo pacchetto è destinato a tutte quelle piccole imprese che necessitano di mantenere un buon livello di persistenza dei dati e del servizio.
- *vSphere Standard* – fornisce una soluzione entry-level per le medie imprese, che hanno l'obiettivo di minimizzare i costi per l'hardware e di accelerare il deploy delle applicazioni. Questa suite comprende tutti i servizi base per virtualizzazione e storage, oltre a quelli per la High Availability e la migrazione delle VM. La licenza deve essere acquistate per ogni processore utilizzato (con un massimo di 6 core e 256Gb di RAM per host).
- *vSphere Advanced* – questa versione aggiunge alcune funzionalità a vSphere Standard in merito al mantenimento della disponibilità del servizio, è possibile infatti utilizzare gli strumenti per la fault tolerance, data recovery e funzioni di sicurezza più avanzate. Questa versione supporta processori con al massimo 12 core ciascuno e host con al massimo 256GB di RAM.
- *vSphere Enterprise* – oltre alle caratteristiche delle due versioni precedenti, si aggiungono funzionalità richieste per proteggere i dati da guasti e di automatizzare la gestione delle risorse e dei consumi.
- *vSphere Enterprise Plus* – aggiunge alla versione Enterprise dei tool per poter gestire la propria cloud e le applicazioni in esecuzione su essa in maniera ancora più semplice; la licenza di questo prodotto consente di utilizzare vSphere su processori con al massimo 12 core e su host con qualsivoglia quantità di RAM.

Considerazioni

vSphere risulta un prodotto molto adatto alle piccole e medie imprese, le quali sono spesso costrette ad affrontare le stesse problematiche IT delle grandi aziende, ma con possibilità di spesa meno elevata. La soluzione proposta offre alle piccole organizzazioni il modo più economico per creare un'infrastruttura *IT* capace di offrire l'elevata disponibilità, l'affidabilità e la semplicità di gestione delle operazioni di cui questo tipo di imprese necessitano. Per prima cosa si possono ridurre sensibilmente le spese per l'acquisto di nuovi server e per la loro gestione grazie al cosiddetto *consolidamento dei server*, ossia è possibile concentrare il carico di lavoro su meno macchine, riducendo anche i consumi energetici. Alcuni utilizzatori di vSphere hanno ottenuto rapporti di consolidamento di 15 a 1, potendo così dedicare le macchine libere a nuove applicazioni per i propri clienti. La facilità di installazione e gestione del sistema garantisce anche un incremento della produttività dello staff IT all'interno dell'impresa; si potrà quindi dedicare più tempo allo sviluppo di soluzioni più utili per l'azienda, senza doversi preoccupare di altre attività di gestione necessarie se le applicazioni venissero implementate su server fisici. Tutti i sistemi di backup automatizzato e sicuro aiutano a proteggere applicazioni e dati critici indispensabili per la continuità aziendale grazie alla possibilità di effettuare la manutenzione

dell'hardware senza downtime; inoltre un ambiente virtualizzato si presta molto più facilmente ad essere utilizzato come banco di prova per nuovi applicativi.

Anche le compagnie che dispongono di grandi data center possono trarre dei vantaggi dall'utilizzo di questa piattaforma, minimizzando i consumi energetici (con il consolidamento dei server), ma anche sfruttando l'elasticità caratteristica dei sistemi virtualizzati, che garantisce una maggior facilità di gestione dell'intero data center. vSphere è un sistema cloud abbastanza statico, infatti ogni macchina viene considerata un'entità a sé stante (con l'eccezione di vApps), non si può in effetti parlare di IaaS, infatti lo scenario caratteristico per questo tipo di applicazione è costituito da una serie di macchine virtuali poco dinamiche, che non vengono accese e spente frequentemente. Il sistema risulta quindi estremamente utile per aziende con risorse IT già consolidate, che conoscono già quali sono le propire esigenze; vSphere però è un'ottima base per lo sviluppo di un servizio di IaaS, implementato dalla stessa VMware con il software vCloud Director che si colloca esattamente al livello superiore di vSphere.

6.6 vCloud

VMware ha sviluppato recentemente una suite di programmi denominata *vCloud*, per sfruttare tutte le potenzialità tipiche di un sistema cloud, quali efficienza, flessibilità e costi ridotti. L'obiettivo principale di queste applicazioni è per prima cosa quello di minimizzare i costi IT delle aziende, sia per quanto riguarda l'acquisto che per la gestione dei macchinari. L'adozione di standard condivisi consente di eseguire le proprie VM direttamente nel proprio data center che implementa i servizi cloud, o sfruttando risorse di cloud provider esterni.

6.6.1 vCloud Director

vCloud Director permette alle aziende di costruire delle cloud private sicure e affidabili, raggruppando le proprie risorse in Datacenter Virtuali, che saranno poi assegnati ad utenti interni e gestiti tramite un'interfaccia web-based; questi data center sono da considerarsi completamente isolati e indipendenti l'uno dall'altro. Così facendo si ottiene una completa astrazione tra le risorse fisiche e quelle che ogni utente sta utilizzando. I vantaggi di questo approccio sono innanzitutto un maggiore utilizzo dell'hardware e, di conseguenza, una diminuzione del numero di risorse fisiche richieste; inoltre si ha anche una grande flessibilità, dato che risulta estremamente semplice e veloce aumentare o diminuire il numero di risorse assegnate a ogni virtual data center.

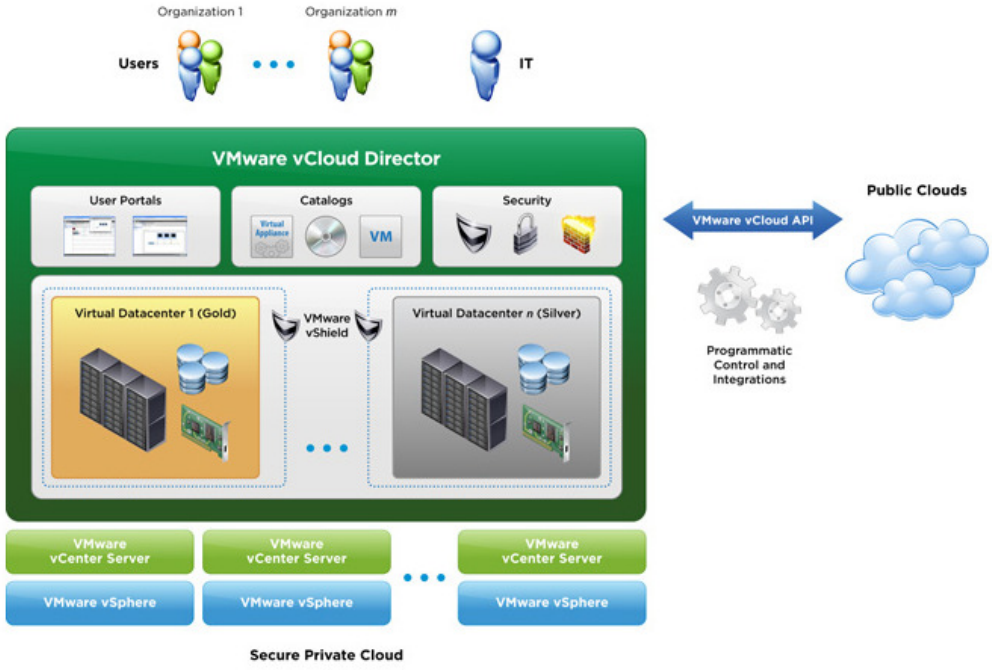


Figura 22 Architettura di VMware vCloud Director

A livello utente vCloud Director consente di utilizzare risorse secondo un paradigma self-service, è sufficiente infatti scegliere quale tipo di virtual data center si necessita (scelto tra alcuni modelli proposti dall'amministratore) e renderlo attivo in pochissimo tempo, grazie all'agilità del paradigma cloud.

Risorse Fisiche

vCloud Director può utilizzare risorse interne, interagendo con infrastrutture gestite da vSphere, sfruttandone tutte le potenzialità per la gestione delle VM, dello storage e del networking; inoltre la presenza di vShield consente di mantenere isolati i virtual data center uno dall'altro e di garantire ottimi livelli di sicurezza dei dati. Dato che vCloud Director utilizza degli standard aperti (vCloud API) per mantenere il massimo di flessibilità nella gestione del sistema e per poter costruire delle cloud ibride; se il numero di risorse fornite da vSphere non fosse sufficiente, è possibile includere parte delle risorse fornite da dei cloud providers compatibili e gestirle come appartenessero alla propria cloud privata.

Licenze e prezzi

vCloud Director viene venduto ad un costo di circe 3750$ annui, con la possibilità di gestire al massimo 25 VM. È necessario però possedere anche licenze dei componenti per la gestione hardware, quali vSphere (edizione Enterprise Plus) e vCenter Server.

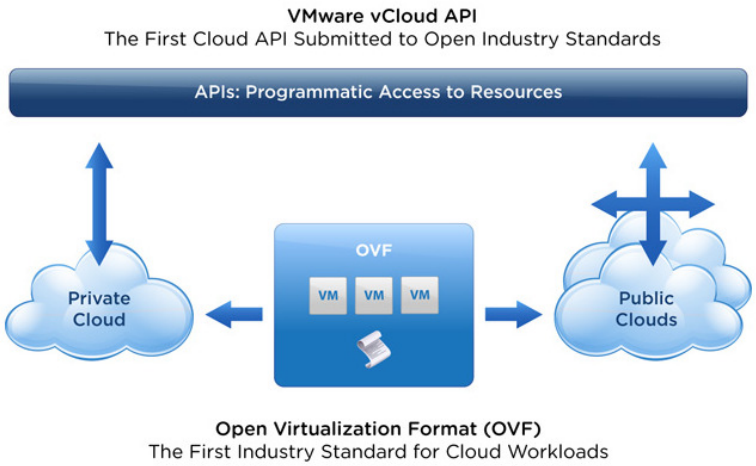


Figura 23 vCloud supporta lo standard Open Virtualization Format (OVF)

Considerazioni

La flessibilità offerta da *vCloud Director* è sicuramente uno dei suoi punti di forza principali, dato che garantisce la possibilità di avviare un data center in pochissimo tempo, senza nemmeno dover contattare un amministratore di sistema. Inoltre tutta la struttura si basa su prodotti già ben consolidati negli anni, quali vSphere e tutti i tool ad esso legati (VMware ESX, vShield , vStorage ecc.). L'utilizzo da parte di VMware di standard aperti per la gestione della cloud può essere un punto di forza molto significativo rispetto ai propri concorrenti, in particolare Amazon EC2, che invece utilizza standard proprietari; molte compagnie infatti potrebbero scegliere di non fare la scelta di utilizzare tali standard, al fine di evitare problemi di *lock-in* futuri. Inoltre il sistema cloud di VMware consente di ammortizzare le spese già sostenute da parte delle aziende, soprattutto se queste hanno già implementato sui propri server prodotti VMware; con vCloud director, un'azienda può continuare a sfruttare le proprie risorse, integrandole con quelle fornite da altri provider, dovendo sostenere spese molto contenute per passare al sistema cloud sviluppato da VMware. vCloud Director non si adatta molto bene a sistemi sviluppati per essere autoscalabili, infatti la procedura di copia di una VM risulta abbastanza lenta, rendendo inapplicabile tale paradigma.

7. AMAZON AUTOSCALING E ELASTIC LOAD BALANCING

Amazon fornisce un servizio per gli utenti di EC2 che consente di creare dei cluster di istanze autoscalabili [15]: un cluster è definito dall'utente come un insieme di istanze, il cui numero è variabile a seconda del carico di lavoro da eseguire in ogni momento. Il service provider si occuperà di gestire tutte le operazioni di aggiunta e rimozione di macchine virtuali, a seconda delle necessità, in maniera totalmente automatica. Una funzionalità di questo tipo può rivelarsi estremamente utile per tutti quei clienti le cui applicazioni sono soggette a picchi di richieste più o meno prevedibili. Sfruttando il servizio "Autoscaling" è possibile rendere disponibile un servizio che è in grado di poter rispondere a un numero di richieste pressoché illimitato; il sistema infatti nel momento in cui la quantità di risorse utilizzate superasse un certo limite, aggiungerà una nuova istanza, rendendo così il sistema in grado di soddisfare tutte le richieste degli utenti. Il servizio *Autoscaling* garantisce anche a chi lo utilizza di minimizzare i costi del proprio sistema, infatti la quantità di risorse allocate in ogni istante sono dimensionate al carico attuale dell'applicazione, evitando quindi all'utente di pagare per risorse inutilizzate e, nello stesso momento, di evitare fenomeni di underprovisioning.

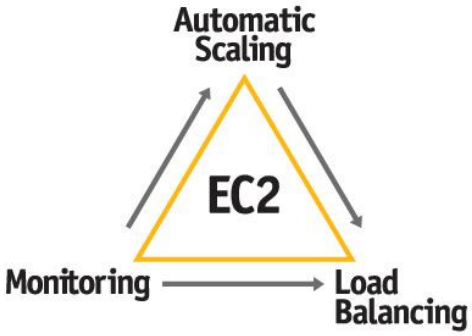


Figura 24 Autoscaling deve collaborare con altri servizi Amazon durante la sue esecuzione

Il funzionamento di *Amazon Autoscaling* è molto semplice: il sistema monitora periodicamente l'utilizzo di risorse delle macchine virtuali in esecuzione, in accordo con i parametri definiti dall'utente. Se il *monitor* verifica che una determinata soglia è stata superata, avvia una nuova istanza e, se richiesto, la inserisce tra gli indirizzi di smistamento di traffico del *load balancer* (anch'esso un servizio

Amazon); può succedere anche la situazione opposta, ossia quando le risorse utilizzate sono inferiori a una determinata soglia, il sistema sceglie un'istanza di quelle attive e le invia il segnale di spegnimento, aggiornando di conseguenza anche il load balancer.

Autoscaling interagisce con altri servizi forniti da Amazon: *Elastic Load Balancing*, che fornisce un load balancer in grado di gestire il bilanciamento di carico tra le istanze del cluster e di garantire la persistenza delle sessioni attive, *CloudWatch*, per monitorare le risorse utilizzate dalle macchine virtuali e EC2 per l'esecuzione vera e propria delle macchine.

7.1 Elementi Principali di Amazon Autoscaling

Per poter utilizzare Amazon Autoscaling è necessario definire una serie di componenti i cui ruoli sono fondamentali nell'esecuzione del sistema:

- *Autoscaling Group*: è la rappresentazione di un'applicazione che viene eseguita su una serie di istanze EC2, questo gruppo (cluster) di istanze è in grado di modificare il numero di macchine attive a seconda delle richieste in base alle risorse richieste, oppure tale numero può essere impostato in maniera statica.
- *Launch Configuration*: contiene tutti i parametri necessari per il lancio delle nuove istanze, un autoscaling group deve obbligatoriamente essere collegato ad una launch configuration. È possibile collegare una sola configurazione alla volta, però questa si può aggiornare in qualsiasi momento. Tutte le istanze avviate dopo l'aggiornamento di una launch configuration saranno eseguite seguendo i parametri scelti dall'utente.
- *Trigger*: è il componente che contiene tutte le informazioni necessarie per il monitoring delle istanze e per poter scegliere quando si dovrà attivare una nuova istanza o terminarne una in esecuzione
- *Scaling Activity o Scaling Operation*: sono tutti quei processi che apportano dei cambiamenti agli elementi dell'autoscaling group, tali operazioni richiedono tempi di esecuzione abbastanza lunghi (avviare un'istanza su EC2 può richiedere qualche decina di minuti). È importante tenere conto dei tempi di esecuzione di questi processi per evitare stati inconsistenti del cluster o eseguire operazioni con le tempistiche sbagliate.
- *Availability Zones*: su Amazon è possibile dividere il proprio autoscaling-group su più availability-zones, ognuna delle quali rappresenta un data center in diverse località del mondo. Utilizzare questo servizio consente quindi di ottenere gradi di sicurezza e disponibilità del servizio molto elevati.

Per le definizioni e la modifica di ognuno di questi elementi ogni utente può interagire con Amazon mediante dei comandi da shell o con invocazioni a servizi web, mediante delle api REST o SOAP. Per una descrizione dettagliata di tutti i parametri che si possono utilizzare si veda la documentazione consultabile sul sito dei web services di Amazon.

7.2 Utilizzo tipico di Amazon Autoscaling

I sistemi che maggiormente sono in grado di sfruttare tutte le potenzialità di Amazon Autoscaling sono quelli in cui molteplici copie della stessa applicazione vengono messe in esecuzione in modo da dividere il traffico di richieste in ingresso. Questa distribuzione del carico è trasparente all'utente finale, che interagisce con l'applicazione come se fosse eseguita su una singola macchina remota. Il sistema di Amazon provvede a monitorare lo stato delle istanze mediante processi interni, che verificano innanzitutto se le istanze considerate sono effettivamente attive, riavviandole se queste terminassero in maniera improvvisa; così facendo viene garantita la presenza di un numero di istanze attive sempre coerente con quanto previsto dall'utente. Oltre al servizio di *healthcheck* è sempre attivo un processo di *monitoring,* che viene definito descrivendo dei Trigger, i quali contengono tutte le informazioni necessarie per effettuare le scelte di scaling-up e scaling down. Un trigger contiene non solo informazioni su quale parametro controllare e sulle soglie da utilizzare, ma anche dei parametri che descrivono per quanto tempo queste soglie devono essere superate, in positivo o in negativo, per procedere con le scaling operation richieste. È importante definire anche questi ultimi parametri al fine di evitare che picchi isolati facciano avviare delle procedure non necessarie, che comporterebbero delle spese aggiuntive per l'utente senza fornirgli nessun vantaggio prestazionale. I *trigger* sono uno dei componenti più importanti del sistema di autoscaling di Amazon, vi sono due parametri fondamentali che ogni utente deve impostare secondo le proprie esigenze: *Period*, che determina l'intervallo di tempo che intercorre tra una misurazione e l'altra, e *BreachDuration*, che definisce per quanto tempo il valore riscontrato deve restare oltre la soglia perché venga attivato il segnale di avvio della scaling operation. Il monitoring delle risorse viene affidato al componente Amazon *CloudWatch*, che è in grado i monitorare la percentuale di utilizzo della CPU, la banda in ingresso e uscita misurata sull'interfaccia di rete e i dati relativi al numero di accessi in lettura e scrittura sul disco. La varietà di queste metriche consente all'utente di scegliere quella che più si adatta alla propria applicazione.

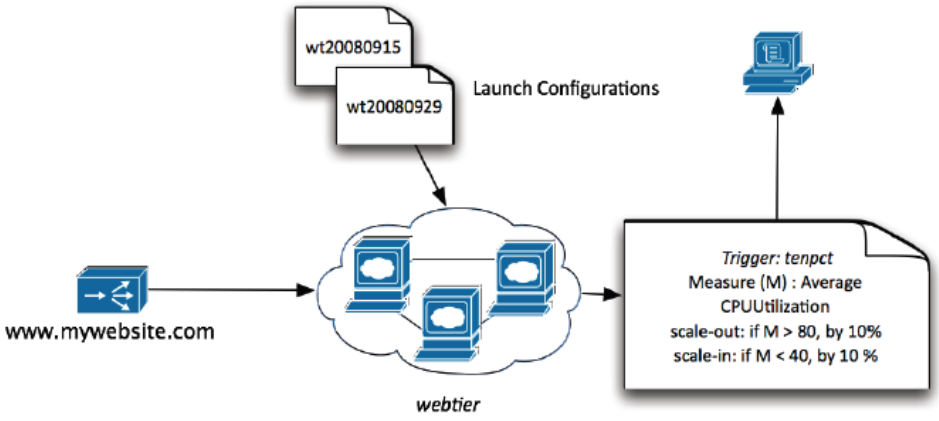


Figura 25 Schema di Funzionamento di Amazon Autoscaling

Il trigger, una volta verificato che tutte le soglie definite sono state superate, richiede le informazioni della Launch Configuration e avvia una nuova istanza verificando che questa completi correttamente le

procedure di startup. Autoscaling può essere associato ad un'istanza di *Elastic Load Balancer*, che verrà automaticamente aggiornata in base alle istanze attive dell'Autoscaling Cluster.

7.3 Eastic Load Balancer e Sticky Sessions

L'utilizzo del servizio Autoscaling in collaborazione con *Elastic Load Balancer* risulta molto naturale e facile da implementare per l'amministratore dell'applicazione, visto che si permette di ottenere un bilanciamento del carico di richieste su tutte le istanze attive. Elastic Load Balancer oltre a gestire le richieste in arrivo, effettua dei controlli periodici sullo stato delle istanze a lui associate, se queste non fossero più attive non inoltrerà verso quest'ultime altre richieste. Il load balancer di Amazon è rappresentato da un nome DNS e da un insieme di porte sulle quali ascolta, oltre ad altri parametri quali le availability-zones verso le quali invierà le richieste; questo particolare accorgimento di non associare un IP fisso ad un'istanza del load balancer permette allo stesso ELB di poter essere scalabile, sfruttando delle procedure interne gestite dai sistemi Amazon. Si è visto effettuando alcuni test di performance che le richieste a cui il load balancer riesce a dare risposta aumentano mentre aumenta il carico di richieste [56].

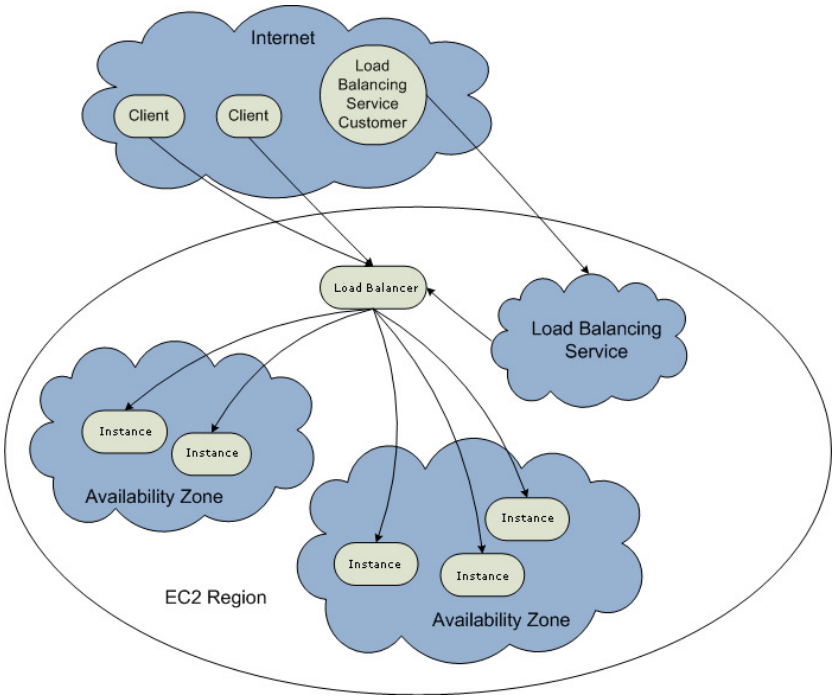


Figura 26 Schema di funzionamento di Amazon Elastic Load Balancer

Un'ultima caratteristica del load balancer di Amazon è la possibilità di mantenere la persistenza della sessioni, rendendole "sticky", funzionalità molto importante visto che nella maggior parte delle applicazioni utilizzate oggi richiedono di poter interagire direttamente con l'utente, personalizzando il risultato delle richieste per ognuno di essi. Per poter garantire tali servizi è quindi necessario mantenere la persistenza delle sessioni, ossia fare in modo che le comunicazioni provenienti da un utente vengano inoltrate sempre alla stessa istanza; così facendo si può mantenere una continuità nella comunicazione

facendo in modo che l'utente finale non si renda conto che l'applicazione a cui sta accedendo è in esecuzione su un cluster di istanze il cui traffico è smistato da un load balancer.

Sticky Sessions

Vi sono due livelli di persistenza delle sessioni: il primo è costituito dalla stickiness a *livello di trasporto*, o layer 4 della pila ISO-OSI, che viene implementato analizzando i flussi TCP e inoltrando i pacchetti appartenenti a una stessa comunicazione al server ad essa associato; questo tipo di persistenza è la più semplice da implementare, ma è indicata solo per quelle applicazioni che utilizzano un unica flusso TCP per ogni sessione, come ad esempio SSH, flussi SSL ecc. Se si considerano invece le web applications che espongono servizi dinamici, come pagine web o altri servizi analoghi, si vede come una sessione-utente sia costituita da molteplici connessioni TCP, quindi non basta mantenere un flusso TCP sullo stesso server, ma bisogna anche inoltrare tutte le successive richieste di tale sessione allo stesso nodo, dato che è solo questo server a conoscere tutti i dati relativi alla specifica sessione. Il tipo persistenza appena descritto viene detta stickiness a *livello applicativo*, o layer 7 della pila ISO-OSI, e viene solitamente implementato analizzando i cookie associati alle richieste e alle risposte inviate al server, e utilizzando quante informazioni per scegliere a quale server inviare le richieste in arrivo.

Amazon ha aggiunto il supporto alle *sticky session* a livello 7 nel mese di Aprile 2010 [57]; il servizio è stato integrato in Elastic Load Balancing: sono supportate due policy per la gestione della persistenza delle sessioni:

- *Load Balancer-Generated HTTP cookie*: ELB utilizza un cookie speciale aggiuntivo per inserire le informazioni di cui necessiterà per le richieste successive. Quando il load balancer riceve una richiesta, per prima cosa verifica se è presente un cookie conosciuto, cioè aggiunto in una precedente comunicazione, e ne analizza il contenuto, inoltrando la richiesta in base alle informazioni contenute nel cookie stesso. Se invece non sono presenti cookie del tipo richiesto, il load balancer ne aggiunge uno al messaggio di risposta inviato dal server, questo cookie aggiuntivo verrà poi inviato dal client insieme alle successive richieste. Uno dei parametri che l'utente deve definire è l'expire-time associato al cookie, che determina la durata della sessione a livello applicativo.
- *Application-Generated HTTP cookie*: il load balancer utilizza anche in questa modalità un cookie speciale per associare le sessione con una delle istanze del cluster, ma invece di inserire un cookie ed associargli un expire time, analizza le operazioni che il server effettua sui cookie generati per gestire la sessione. Il load balancer inserirà un nuovo cookie solamente se l'applicazione stessa ha aggiunto alla risposta il cookie di sessione; il sistema inoltre analizza anche eventuali messaggi inviati dal server contenenti istruzioni per rimuovere tale elemento. Se l'applicazione rimuove il cookie di sessione esplicitamente, Amazon Elastic Load Balancer interpreta questa operazione come un'esplicita terminazione della sessione, rimuovendo il cookie aggiuntivo per la stickiness.

Per abilitare la persistenza delle sessioni a livello applicativo è necessario definire una stickiness policy e associarla al load balancer che si vuole utilizzare; si deve definire il nome da associare alla policy, il load balancer su cui applicarla e l'expire-time da assegnare ai cookie aggiuntivi. Ci sono due diversi tipi di richieste per creare i due tipi di policy supportati: `CreateLBCookieStickinessPolicy`, per i loadbalancer-generated cookie, e `CreateAppCookieStickinessPolicy`, per gli application-generated cookie; nella seconda richiesta si deve definire, oltre agli altri parametri, il nome del cookie generato dall'application server, che il load balancer dovrà monitorare.

```
elb-create-app-cookie-stickiness-policy my-load-balancer -p my-appcookie-lb-policy -c my-cookie
```

È possibile aggiornare in qualsiasi momento la stickiness policy associata a un load balancer, mediante una richiesta del tipo `SetLoadBalancerPoliciesOfListener`, in cui si deve fornire il nome della nuova policy da utilizzare, oltre al nome e alla porta del load balancer.

```
elb-set-lb-policies-of-listener example-lb --lb-port 80 -policynames my-app-cookie-lb-policy
```

La gestione delle sessioni a livello applicativo garantisce un buon livello di personalizzazione e risulta quasi completamente trasparente all'utente finale, che visualizzerà solo un cookie aggiuntivo per ogni sessione attiva.

Limitazioni di Elastic Load Balancing

Il sistema di load balancing proposto da Amazon presenta alcune limitazioni in determinati scenari di utilizzo, infatti non è sempre possibile mantenere le sessioni sticky se non si soddisfano alcuni requisiti necessari.

- ELB non supporta la persistenza delle sessioni per comunicazioni *criptate*, come i flussi di dati HTTPS; tale limitazione è dovuta al fatto che il sistema deve poter analizzare i cookie trasmessi con i messaggi HTTP, nel caso si utilizzi una connessione cifrata, il load balancer non ha accesso a questi dati, di conseguenza non è in grado di scegliere a quale application server inoltrare le richieste. Per poter bilanciare flussi di dati criptati è necessario aggiungere al load balancer applicativi aggiuntivi, in grado di analizzare correttamente i dati criptati, come *aiCache*.
- Un'altra funzionalità che non è supportata dal load balancer di Amazon e dal servizio di Autoscaling riguarda la *Scale-Down proof Stickiness* [58]. Il sistema infatti non si occupa di mantenere la persistenza delle sessioni durante le operazioni di *Scale-down* (spegnimento e rimozione di una macchina virtuale) di un cluster di istanze. Viene ignorato il fatto che ci potrebbero essere delle sessioni attive associate alla macchina virtuale in fase di spegnimento, può succedere infatti che alcuni utilizzatori del servizio vedano terminare le proprie sessioni improvvisamente. Questa situazione può accadere regolarmente se si utilizzano cluster di istanze il cui numero si adatta in base al carico di richieste e il numero di richieste è molto variabile. Il sistema di Amazon è in grado di gestire in maniera corretta tutte le operazioni di

scaling-up del numero delle macchine, inserendo la nuova istanza tra le entry del load balancer una volta che questa ha completato le proprie operazioni di avvio. Quando invece si deve effettuare il procedimento inverso, Autoscaling rimuove dal load balancer l'istanza che verrà spenta e ne inizia la procedura di shutdown; così facendo è possibile che gli utenti le cui richieste erano state assegnate all'application server appena rimosso vedano terminare improvvisamente la loro sessione, dato che le loro richieste saranno inoltrate ad un server diverso da quello precedente. L'unico modo di risolvere questa problematica è quello di modificare la politica di gestione delle sessioni al livello degli application server: si deve fare in modo che i dati delle sessioni siano salvati su un supporto condiviso, come un database unico accessibile dalle varie istanze, o utilizzare un sistema di replicazione di tali dati su tutti gli application server.

7.3.1 Esempio di configurazione di un Autoscaling-Cluster

Per definire un Autoscaling cluster è sufficiente effettuare le operazioni necessarie per impostare i parametri di tutti gli elementi fondamentale per l'avvio del sistema [59]:

- Se necessario, si deve *definire un load balancer*, creando un'istanza di Elastic Load Balancer, indicando informazioni quali porta di ascolto, availability-zone e porta su cui verrà effettuato il forward delle richieste alle istanze. Con il comando:

  ```
  elb-create-lb MyLoadBalancer --headers --listener "lb-
  port=80,instance-port=8080,protocol=HTTP" --availability-zones us-
  east-1a
  ```

- Si deve poi definire una *LaunchConfiguration,* da utilizzare per l'avvio delle istanze, in questo componente vanno definiti l'id dell'immagine da avviare e il tipo di istanza, che indica quante risorse verranno allocate per quest'ultima dal sistema di Amazon:

  ```
  as-create-launch-config MyLaunchConfiguration --image-id myAMI--
  instance-type m1.small
  ```

- A questo punto è possibile creare l'Autoscaling Group, indicando il load balancer e la launch configuration precedentemente indicati e il numero massimo e minimo di istanze che possono essere in esecuzione:

  ```
  as-create-auto-scaling-group MyAutoScalingGroup --launch-
  configuration MyLaunchConfiguration --availability-zones us-
  east-1a --min-size 2 --max-size 20 --load-balancers
  MyLoadBalancer
  ```

- Come ultima operazione di deve creare un *Trigger*, in cui vanno indicati la metrica da misurare, il tipo di dati che essa ritorna, il periodo di misurazione e la *BreachDuration*, oltre ai valori di soglia massima e minima:

  ```
  as-create-or-update-trigger MyTrigger --auto-scaling-group
  MyAutoScalingGroup --namespace "AWS/EC2" --measure CPUUtilization --
  statistic Average -dimensions
  "AutoScalingGroupName=MyAutoScalingGroup" --period 60 --lower-
  threshold 40 --upper-threshold 80 "--lower-breach-increment=-1" "--
  upper-breach-increment=1" --breach-duration 600
  ```

Il cluster di istanze è ora operativo, il sistema inizierà a svolgere le prime scaling operation per avviare le macchine virtuali necessarie per soddisfare il vincolo sul numero minimo di istanze attive, successivamente le operazioni verranno effettuate quando il carico di lavoro supererà le soglie definite dal trigger.

7.3.2 Aspetti pratici dell'utilizzo di Amazon Autoscaling

Una volta impostato correttamente e avviato il cluster di istanze, di deve porre l'attenzione anche a come le istanze che vengono automaticamente avviate e spente dal sistema riescano ad interagire tra loro e a mantenersi costantemente aggiornate. Infatti tutte le macchine virtuali che vengono avviate sono sostanzialmente identiche, dato che la loro immagine di partenza è la stessa. L'amministratore del cluster quindi deve affrontare due problematiche principali:

- Si deve garantire che i dati memorizzati nelle istanze in esecuzioni siano *aggiornati e coerenti* tra loro, al fine di evitare di fornire dati diversi alla richieste che vengono gestite da un'istanza rispetto a quelle che vengono assegnate ad un'altra.
- In alcuni casi è anche necessario che le istanze attive siano consapevoli di quante altre istanze sono presenti e di poter *comunicare* con esse. L'amministratore deve fare in modo che una macchina, una volta avviata possa comunicare la propria presenza alle altre del cluster, per poter iniziare a collaborare con queste.

Sincronizzazione dei dati

Quasi la totalità delle applicazioni che si possono prestare ad essere utilizzate su un cluster di Amazon Autoscaling devono avere degli applicativi installati, con annessi dei dati che dovranno essere elaborati. Nella maggior parte dei casi le applicazioni installate sono le stesse su tutte le istanze e non vengono mai modificate, se non in casi rari, come ad esempio l'aggiornamento a una versione più recente del software, mentre i dati ad esse collegati possono essere modificati molto più frequentemente. Per questo motivo è conveniente nel caso generale utilizzare un'immagine per l'istanza EC2 che contiene già l'applicativo richiesto, mentre i dati devono essere salvati su supporti differenti, come Amazon EBS o S3.

Adottare una soluzione come quella appena descritta consente di non dover modificare continuamente l'immagine dell'istanza per mantenere aggiornati i dati modificati frequentemente, ma permette di mantenere l'installazione dei propri applicativi sull'immagine, il che facilita le procedure di startup, evitando di dover rieseguire ogni volta le procedure di installazione e configurazione del software.

Use Case: cluster di application server tomcat

Una delle applicazioni che meglio si presta ad essere eseguita su un cluster autoscalabile è sicuramente una web application, ad esempio implementata su server Apache Tomcat. Lo sviluppatore dell'applicazione può implementare la sua applicazione utilizzando diverse immagini EC2:

- Una serie di immagini che si occupano di gestire i dati memorizzati su un *database*, ad esempio MySQL o Microsoft SQL-server
- Un cluster di istanze contenenti *l'application server*, che si occuperanno di interagire con gli utenti
- Il *load balancer* utilizzato per smistare il traffico può essere quello fornito da Amazon, che si integra bene con gli altri servizi che verranno utilizzati.

In questa architettura [60] il load balancer inoltrerà le richieste agli application server, che dovranno interrogare il server SQL, se necessario, elaborare i dati in base alle richieste specifiche degli utenti e inviare le risposte con i dati risultanti.

Sicuramente in questa applicazione per far fronte a picchi di richieste, sarà necessario aumentare il numero delle istanze che eseguono gli application server, in modo da evitare di non riuscire a soddisfare tutte le richieste pervenute. In questa situazione si deve definire un autoscaling cluster, fornendo tutti i parametri sulle risorse da monitorare e sulle soglie da controllare. Oltre a questi parametri si deve definire la launch configuration che il sistema deve utilizzare ogni volta che deve avviare una nuova istanza, è importante poter definire una launch configuration che renda le istanze in grado di essere operative autonomamente e di mantenere i dati dell'applicazione sempre aggiornati. Considerando le immagini che hanno il ruolo di application server, l'amministratore deve valutare come gestire le procedure di *startup* delle istanze, gli aggiornamenti del software e di come mantenere i dati dell'applicazione sempre aggiornati. È evidente che conviene modificare una delle immagini fornite da Amazon, scegliendo il sistema operativo che più si addice all'applicazione che si vuole utilizzare e modificarla: conviene innanzitutto installare il software, effettuando il tuning dei settaggi. Un'altra modifica che conviene fare è quella di inserire negli script di avvio della macchina, con un richiamo ad uno script personalizzato, che permetterà di eseguire le operazioni richieste dall'amministratore, come ad esempio quella di recuperare i dati dell'applicazione, successivamente verrà descritto nel dettaglio un esempio pratico che si addice a questa operazione.

A questo punto si ha un'immagine EC2 che contiene il software richiesto ed esegue, non appena avviata, le operazioni definite dall'amministratore per ottenere i dati dell'applicazione; basta ora preparare l'immagine (fare il "bundling") e caricarla sui server Amazon, in modo da poterla usare per la launch configuration dell'autoscaling cluster che è stato avviato.

In questo modo si ottiene una separazione netta tra il software, preinstallato sull'immagine dell'istanza, e i dati, che invece devono essere recuperati in altro modo; una soluzione così organizzata richiede il re bundle dell'immagine solamente quando si vuole effettuare un aggiornamento delle impostazioni o della versione del software, evento abbastanza raro.

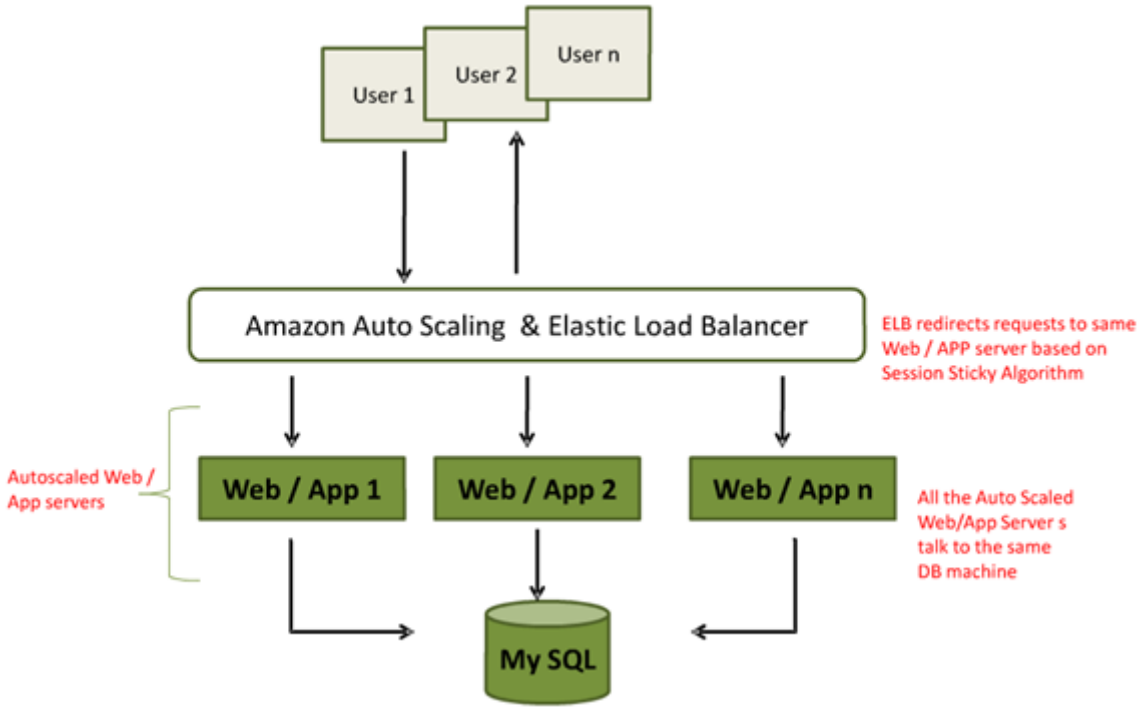


Figura 27 Architettura standard utilizzata per le web application

Per quanto riguarda i dati dell'applicazione, la scelta di mantenerli sull'immagine dell'istanza si rivela non molto producente, basti pensare che ad ogni minimo cambiamento bisognerebbe modificare l'intera immagine e effettuare un nuovo upload sui server di Amazon. Si deve quindi utilizzare una soluzione diversa: una delle più utilizzate è quella di creare un bucket, l'analogo di una cartella, su *Amazon S3*, all'interno del quale vanno memorizzati i dati dell'applicazione. Ogni istanza, nel proprio script di boot, accederà a questa sorta di repository remoto mediante tool come *s3cmd* o *s3rsync*. Questa soluzione è molto funzionale nello scenario descritto, infatti un'istanza, una volta avviata, non deve fare altro che eseguire uno script che effettua il download, o la sincronizzazione, con i dati presenti sul bucket S3. Utilizzando tool come s3rsync si può anche garantire che i dati vengano mantenuti sincronizzati con quelli del bucket S3 in ogni istante. Se l'amministratore volesse effettuare un aggiornamento dei dati dell'applicazione è sufficiente che lo effettui sui dati memorizzati su S3, saranno poi le istanze a sincronizzarsi in maniera totalmente autonoma con le versioni aggiornate dei file modificati.

Istanze Parametrizzate e Script di Boot

Il servizio EC2 di Amazon quando si avvia una nuova istanza, consente di poter associare ad essa dei dati, in formato testo o su file, che costituiscono l'*user-data* associato all'istanza. Questi dati sono accessibili solamente dall'interno della macchina virtuale stessa, contattando un determinato url, che nel caso di EC2 è: http://169.254.169.254/1.0/user-data. Impostare quali dati inviare come user data ad un'istanza è molto semplice, se si utilizzano i tool da linea di comando è sufficiente utilizzare il costrutto `-d <user-data>` o `-f <user-data file>` del comando ec2-run-instances.

Questa funzionalità è estremamente utile per automatizzare tutte le operazioni che devono essere effettuate dopo l'avvio dell'istanza, infatti è possibile inviare o dei parametri in formato testo, che gli

script installati sulla macchina possono riconoscere e eseguire de operazioni associate a tali valori, oppure fornire direttamente un file zip contenente gli script da eseguire. Se si sceglie la seconda opzione, con opportuni accorgimenti, è possibile utilizzare la stessa immagine con script in boot diversi [61], ottenendo una grande flessibilità dell'intero sistema. L'operazione chiave che deve essere effettuata per poter lanciare *script di boot personalizzati* è quella di modificare l'ultimo script che il sistema esegue in fase di avvio: nei sistemi linux si può fare riferimento a `/etc/rc.local` . L'idea è quella di scaricare un file di tipo zip passato come user-data, estrarne il contenuto ed eseguire il file autorun.sh, che deve essere presente in tale archivio. È lo script appena citato che conterrà le effettive operazioni da fare durante l'avvio del sistema: nel caso dello scenario precedetemente descritto nel codice di autorun.sh vi saranno delle chiamate ai tool s3cmd e s3rsync per accedere e memorizzare i dati dell'applicazione dal bucket s3.

```
####### These lines go at the end of /etc/rc.local #######

wget HTTP://169.254.169.254/1.0/user-data \
  -O /tmp/payload.zip

# if wget error code is 0, there was no error
if [ "$?" -e "0" ]; then

  mkdir /tmp/payload
  unzip /tmp/payload.zip -d /tmp/payload/ -o

  # if unzip error code is 0, there was no error
  if [ "$?" -e "0" ]; then

    # if the autorun.sh script exists, run it
    if [ -x /tmp/payload/autorun.sh ]; then

      sh /tmp/payload/autorun.sh

    else
      echo rc.local : No autorun script to run
    fi

  else
    echo rc.local : payload.zip is corrupted
  fi

else
  echo rc.local : error retrieving user data
fi
```

Figura 28 Esempio di script di boot personalizzato

Il fatto di utilizzare uno script così generico consente di poter modificare senza difficoltà le operazioni da eseguire durante il boot della macchina. Ad ogni avvio infatti viene scaricato un file, passato come parametro, ed utilizzato il suo contenuto per eseguire le operazioni richieste, in questo modo gli script eseguiti sono sempre aggiornati a seconda delle necessità dell'amministratore. Le modifiche che sono state effettuate al contenuto dell'immagine originale eseguono operazioni molto generiche, che non necessitano di aggiornamenti frequenti, ognuno dei quali richiederebbe un nuovo bundle e upload

dell'intera immagine; per aggiornare gli script da eseguire invece è sufficiente modificare il file passato come parametro al momento dell'avvio della macchina.

Nello script `autorun.sh` non solo si possono inserire istruzioni semplici, quali il download i file o la modifica di qualche parametro software, ma si può utilizzare per effettuare aggiornamenti, installare pacchetti, modificare impostazioni di sistema, ecc. Inoltre essendo gli script passati come parametro ad ogni avvio dell'istanza, non vi sono nemmeno i problemi legati alla modifica e aggiornamento di tali file, basta semplicemente aggiornare l'archivio zip dell'user-data, mentre la parte generica dello script installata sull'immagine della macchina può restare invariata.

8. CONFRONTO TRA SOLUZIONI IAAS E PAAS

Le soluzioni cloud di tipo *Platform-as-a-Service* offrono dei servizi che mettono a disposizione una piattaforma sulla quale gli sviluppatori possono installare e eseguire le proprie applicazioni; il service provider si occuperà di gestire la scalabilità dell'applicazione, allocando in maniera completamente autonoma tutte le risorse necessarie per l'esecuzione delle operazioni richieste. L'utente dovrà poi pagare una tariffa in base alle risorse effettivamente utilizzate, come il tempo di CPU impiegato e la banda occupata. L'infrastruttura che viene utilizzata per implementare effettivamente il servizio non è visibile all'utente, a cui viene data solamente la possibilità di utilizzare la piattaforma caricando le proprie applicazioni, come se queste venissero installate su un application server. Uno sviluppatore deve quindi valutare se sia più conveniente utilizzare una soluzione di tipo Platform-as-a-Service oppure ricreare un sistema con le stesse funzionalità utilizzando un servizio Infrastructure-as-a-Service. È possibile infatti ricreare un sistema che simula il comportamento di un provider PaaS, mediante delle risorse fornite da un service provider IaaS; in questo modo l'utente può rendere la propria applicazione scalabile esattamente come se fosse eseguita su un servizio PaaS, ma senza tutte le limitazioni imposte da questo tipo di soluzione cloud, di cui verranno successivamente elencati alcuni esempi. La differenza sostanziale che si presenta tra una soluzione PaaS e IaaS riguarda la libertà di utilizzo, visto che in un caso lo sviluppatore deve rispettare alcuni vincoli imposti, mentre nell'altro ha il pieno controllo del sistema; a questo però si contrappone un grado di difficoltà maggiore se si vogliono utilizzare le risorse IaaS, infatti l'utente dovrà impostare correttamente le macchine virtuali perché queste possano collaborare tra di loro e gestire correttamente il bilanciamento del traffico tra queste. Inoltre devono essere impostati tutti i parametri che governano le operazioni di Scaling e che vengono utilizzati per decidere quando si deve aggiungere o rimuovere un'istanza dal cluster per far fronte alle variazioni del numero di richieste in arrivo. Nella sezione seguente verranno analizzate le differenze, sia economiche che funzionali, tra due soluzione, una di tipo Paas (Google App Engine) e una IaaS (Amazon EC2).

8.1 Differenze tra Google App Engine e Amazon EC2

Google App Engine mette a disposizione degli sviluppatori una piattaforma in grado di eseguire applicazioni web scritte in linguaggio Java o Python, che l'utente può sviluppare e testare in locale, per poi eseguirle sulla piattaforma fornita dal service provider. Il sistema garantisce che le applicazioni siano completamente scalabili, fornendo agli sviluppatori la possibilità di disporre di risorse virtualmente infinite, per far fronte ad aumenti di richieste anche molto elevati. Il vantaggio di utilizzare il servizio offerto da Google è quello di non doversi preoccupare in alcun modo di come strutturare le proprie applicazioni in modo che possano essere scalabili, sia dal punto di vista della topologia di rete, che per quanto riguarda la gestione del software. Sarà il service provider che si occuperà di effettuare tutte le operazioni necessarie per aumentare la quantità di risorse allocate per una determinata applicazione, qualora vi fosse un aumento del carico di lavoro; queste procedure sono completamente trasparenti sia all'utente finale sia allo sviluppatore, che dovrà solo pagare al provider in base alle risorse consumate dall'esecuzione della propria applicazione.

La piattaforma offerta da Google App Engine è compatibile con le applicazioni Python e Java, tuttavia vengono applicate alcune *limitazioni*; lo sviluppatore non ha quindi un controllo e una libertà totale per l'esecuzione delle proprie applicazioni, ma deve rispettare i vincoli imposti dal service provider come:

- accesso in sola lettura al file system
- possibilità di eseguire solamente codice invocato tramite HTTP request
- le applicazioni Java possono utilizzare solo un sottoinsieme delle classi disponibili nel JRE Standard Edition (The JRE class White List)
- è impossibile creare nuovi thread Java
- supporto limitato alle sole classi "pure-Python", non sono supportati moduli C e Pyrex

La maggior parte di queste limitazioni sono state introdotte per poter gestire al meglio la scalabilità delle applicazioni, oltre per mantenere un elevato livello di sicurezza ed evitare che applicazioni malevole o mal sviluppate possano interferire con l'esecuzione di quelle di altri utenti.

Una soluzione alternativa all'utilizzo di Google App Engine, che lascia più libertà agli sviluppatori delle applicazioni, può essere quella di utilizzare un servizio cloud di tipo IaaS per realizzare un sistema che abbia caratteristiche simili a quelle di un PaaS, per poi utilizzarlo come piattaforma per le proprie applicazioni. Uno dei provider IaaS che fornisce gli strumenti adatti per realizzare un sistema simile a un PaaS è Amazon EC2, infatti si possono utilizzare gli strumenti di monitoring (CloudWatch) e di autoscaling disponibili per poter decidere in quale momento sia opportuno allocare o deallocare le risorse per una determinata applicazione. Inoltre si può utilizzare Elastic Load Balancer per smistare il traffico tra i vari web server, quando questo sia necessario.

Per poter creare un sistema in grado di eseguire un'applicazione Java o Python scalabile, con le stesse funzionalità di *Google App Engine*, si deve avviare un cluster di istanze che eseguono l'application

server adatto e smistare il traffico in arrivo utilizzando *Elastic Load blancer*; quando il numero di risorse supererà una certa soglia, monitorata costantemente da *CloudWatch* (servizio di Monitoring di Amazon), in automatico il servizio *Autoscaling* lancerà una nuova istanza, consentendo al sistema di rispondere a un numero di richieste maggiore. Il sistema così ottenuto avrà le prestazioni e le funzionalità molto simili a quelle che si otterrebbero utilizzando App Engine, ma per realizzarlo viene aggiunto un grado di difficoltà, visto che tutta la gestione del cluster di istanze e di come questo deve reagire ad un aumento del carico di lavoro deve essere effettuata dallo sviluppatore stesso e non dal provider.

Visto che le prestazioni che si ottengono sono molto simili, bisogna anche valutare quanto siano differenti queste due soluzioni dal punto di vista economico, analizzando le diverse tipologie di tariffe proposte e cercando di confrontarle tra di loro, infatti alcuni parametri sono difficilmente comparabili, visto che il livello di astrazione a cui viene fornito il servizio è diverso. Google App Engine applica delle *tariffe a consumo* per tutte le risorse, compreso il tempo di CPU utilizzato, che viene calcolato utilizzando delle procedure interne. Se si utilizza Amazon EC2 invece, le tariffe applicate per l'utilizzo di CPU sono calcolate in base al tempo di esecuzione delle istanze virtuali, anche se i processori assegnati sono in stato idle. Le altre risorse invece vengono fatturate utilizzando sistemi molto simili: il traffico in entrata e uscita viene monitorato e gli utenti pagano in base a quanti dati sono stati trasferiti, solitamente utilizzando due tariffe diverse per i dati in entrata e i dati in uscita. Anche per lo storage dei dati si utilizzano sistemi equivalenti, calcolando la quantità di spazio occupato al mese.

8.2 Analisi e confronto dei costi per l'esecuzione di un' applicazione web Java

Non è possibile effettuare un confronto completamente preciso tra due soluzioni analoghe che utilizzano servizi PaaS (App Engine) e IaaS (EC2), dato che alcuni metodi di fatturazione e alcune funzionalità non coincidono perfettamente, tuttavia con alcuni accorgimenti è possibile valutare come una soluzione possa essere più conveniente rispetto ad un'altra a seconda delle specifiche dell'applicazione che si desidera sviluppare.

Google App Engine: Tariffe

Google App Engine monitora le risorse utilizzate dalle applicazioni di ogni sviluppatore e calcola la quantità di denaro che l'utente deve pagare in base a queste. La tariffe si dividono in cinque diverse voci, ognuna delle quali rappresenta una risorsa.

È facile vedere come molte di queste tariffe siano compatibili con i sistemi di accounting di Amazon EC2, l'unico che invece si discosta dal modello del provider IaaS è rappresentato da "CPU Time". Il sistema calcola la quantità di tempo in cui l'applicazione utilizza le risorse di calcolo ed in base a questi valori calcola il prezzo che l'utente deve pagare; non è quindi immediato poter confrontare tale parametro con la tariffa oraria utilizzata da Amazon EC2

Resource	Unit	Unit cost
Outgoing Bandwidth	gigabytes	$0.12
Incoming Bandwidth	gigabytes	$0.10
CPU Time	CPU hours	$0.10
Stored Data	gigabytes per month	$0.15
Recipients Emailed	recipients	$0.0001

Tabella 11 Tariffe applicate da Google App Engine

Metodi per il confronto delle tariffe di Amazon EC2 e Google App Engine

Un metodo che si può utilizzare per effettuare un confronto tra una soluzione basata su App Engine e una che utilizza EC2 si basa sul *tempo di utilizzo della CPU*: si deve calcolare qual è il numero di macchine virtuali necessarie per effettuare le stesse operazioni che si eseguono su App Engine, valutando poi quali sono i costi relativi non solo all'esecuzione delle istanze, ma anche relativi a tutti gli altri strumenti necessari per gestire il bilanciamento delle richieste. L'obiettivo è quello di poter convertire la quantità di ore-CPU richieste da un'applicazione di App Engine nel numero di istanze EC2 minimo necessario per eseguire lo stesso carico di lavoro, sarà poi sufficiente confrontare la tariffa oraria applicata da Amazon con il costo che comporta l'utilizzo delle ore di CPU calcolate su App Engine per verificare quale soluzione sia la più conveniente.

Per prima cosa si devono definire i parametri medi di ogni richiesta, al fine di poter calcolare accuratamente l'effettiva quantità di risorse utilizzate, in particolare si devono definire:

- la grandezza in Byte di una *richiesta*, che determina la quantità di traffico in ingresso
- la grandezza in Byte della relativa *risposta*, che determina la quantità di traffico in uscita
- lo *spazio occupato* su disco dall'applicazione e dai suoi dati, per valutare le tariffe di storage
- la *quantità di tempo* richiesta dalla CPU per elaborare la risposta, questo è un parametro abbastanza difficile da valutare, ma è fondamentale per poter confrontare le soluzioni dei due provider presi in esame

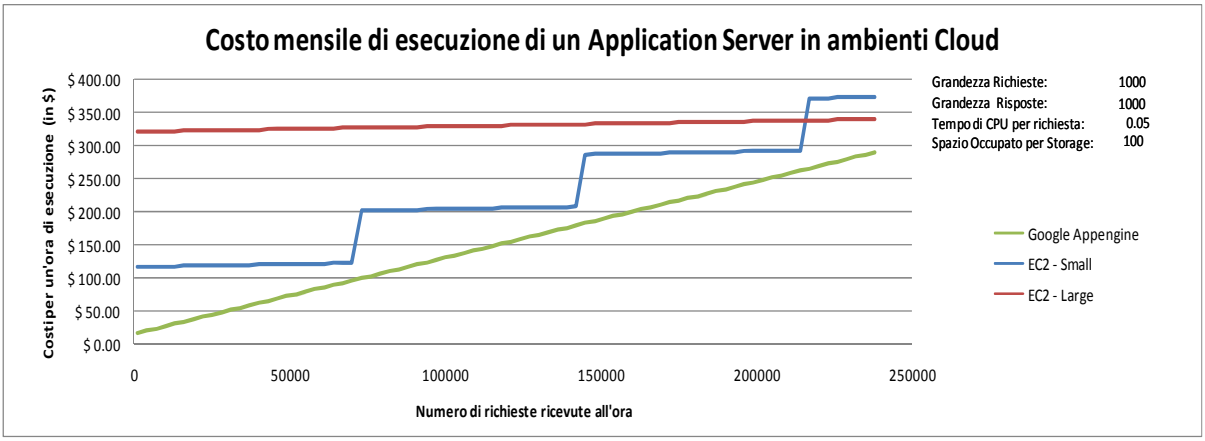
Una volta definita la quantità di richieste che ci si aspetta di ricevere in un'ora, si devono semplicemente calcolare tutti i valori totali delle risorse appena descritte, per ottenere il prezzo orario dell'esecuzione dell'applicazione su App Engine. Per calcolare il costo su EC2 bisogna determinare la quantità di istanze che il sistema dovrà avviare: si deve confrontare il numero di ore-CPU calcolate per App Engine e confrontarlo con le *EC2 Compute Unit (ECU)* della tipologia di istanza che si vuole utilizzare. Una ECU corrisponde a una capacità di calcolo equivalente con un processore di frequenza compresa tra 1.0 e 1-2 GHz, questo valore è compatibile con quello utilizzato per calcolare le ore di CPU su App Engine, che corrispondono al tempo impiegato da un processore di 1.2GHz per eseguire le stesse operazioni. Bisogna però considerare che il tempo di CPU impiegato su App Engine non può considerarsi esattamente uguale a quello utilizzato su EC2, infatti le istanze devono eseguire anche delle

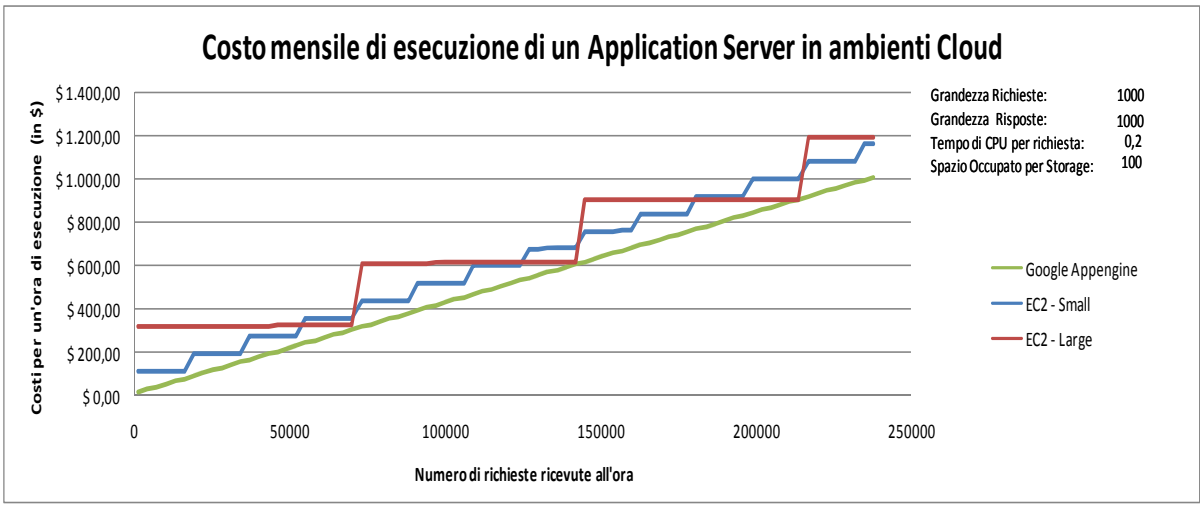
operazioni relative ad altri processi oltre all'application server, basti pensare a tutte le operazioni in background del sistema operativo. Per rendere i due tempi più confrontabili si deve quindi aggiungere una piccola percentuale al tempo richiesto per le istanze EC2, in modo da poter considerare anche i processi esterni al server.

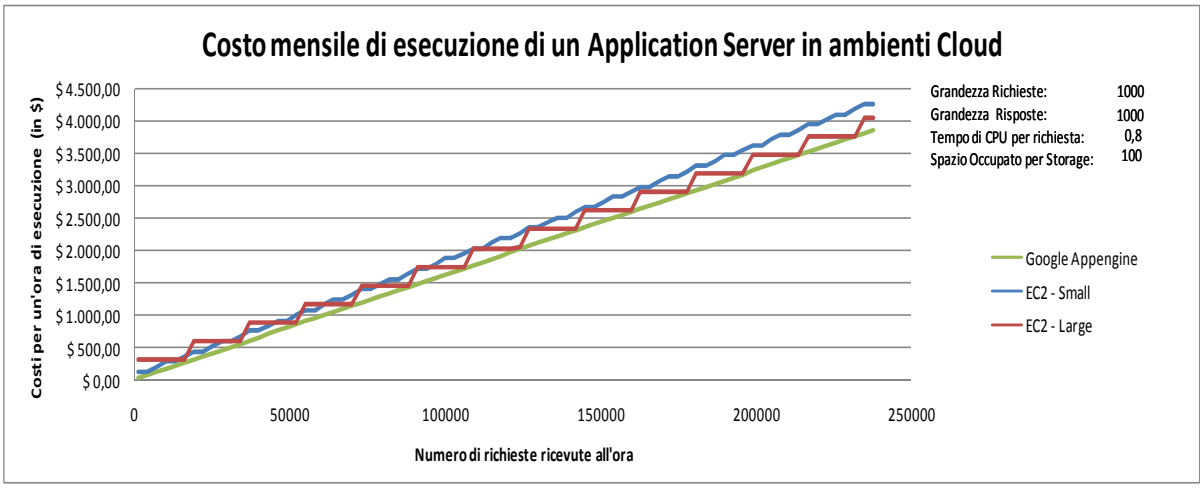
Uno dei parametri più importanti per calcolare il costo di esecuzione su EC2 è il numero di istanze richiesto per poter gestire le richieste in arrivo; determinare il valore di tale parametro non è immediato, infatti dipende da vari fattori, primo fra tutti il tempo di CPU richiesto per elaborare ogni richiesta, ma bisogna anche considerare la quantità di dati in ingresso e in uscita e quanti accessi sia necessario effettuare su disco. Nell'analisi effettuata, per calcolare il numero di istanze richieste su EC2 è stato considerato il parametro relativo alle ore di CPU richieste per App Engine: si divide tale valore per il numero di ECU disponibili per ogni istanza EC2 utilizzata (ad esempio un'istanza Small dispone di 1 ECU, una Large 4 ECU); è necessario prendere sempre l'intero superiore del risultato ottenuto, in modo che siano disponibili tutte le risorse necessarie per rispondere a tutte le richieste in arrivo.

Confronto al variare del tempo di CPU utilizzato

Sono stati valutati i costi mensili di esecuzione mantenendo fissi i parametri relativi alla grandezza delle richieste e delle risposte elaborate, scegliendo per entrambe una dimensione di 1KB, e dello spazio occupato su disco dall'applicazione, fissato a 100GB. Al variare del tempo di CPU richiesto per elaborare ogni richiesta si vede come le soluzioni analizzate tendano ad essere equivalenti, in particolare se si utilizza un tipo di istanza EC2 "large". La soluzione economicamente più vantaggiosa risulta essere, seppur di poco, quella di scegliere Google App Engine. L'andamento a "gradoni" dei grafici relativi alle implementazioni su istanze EC2 sono determinati dal fatto che il prezzo di un'istanza attiva non corrisponde al tempo in cui questa occupa la CPU, ma viene calcolato in base al tempo in cui questa resta attiva, anche se il processore non viene utilizzato.

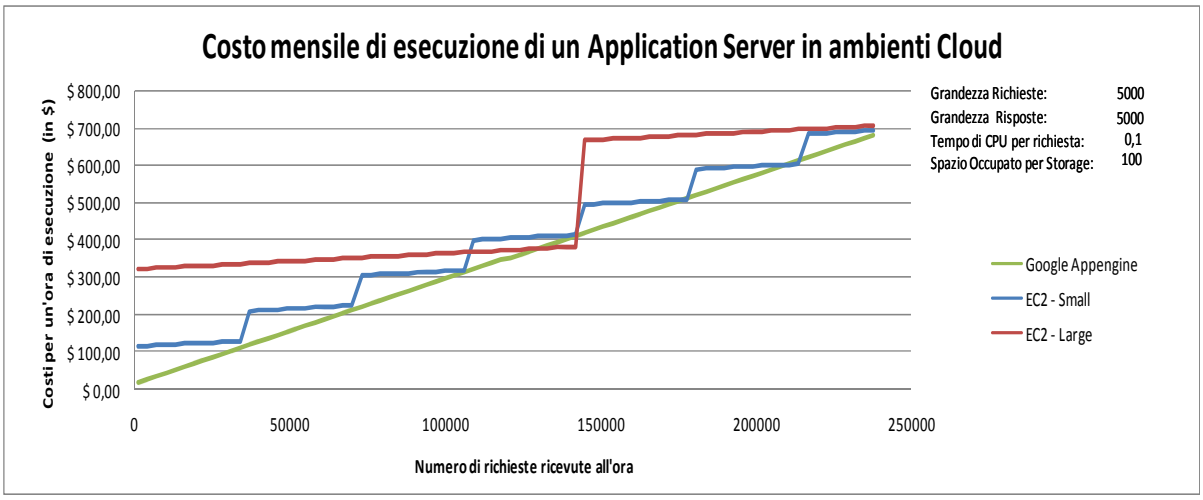


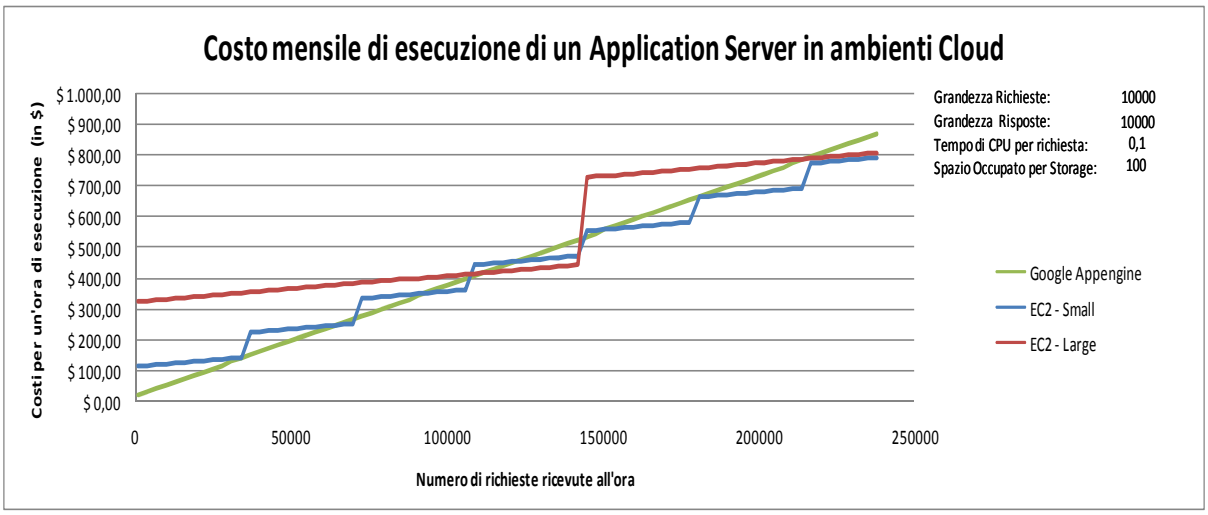


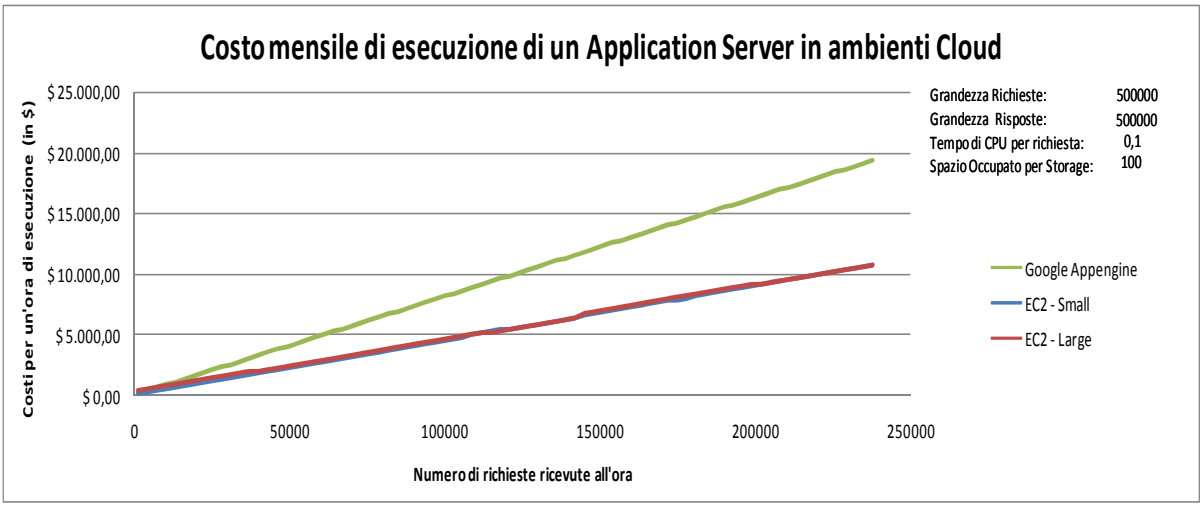


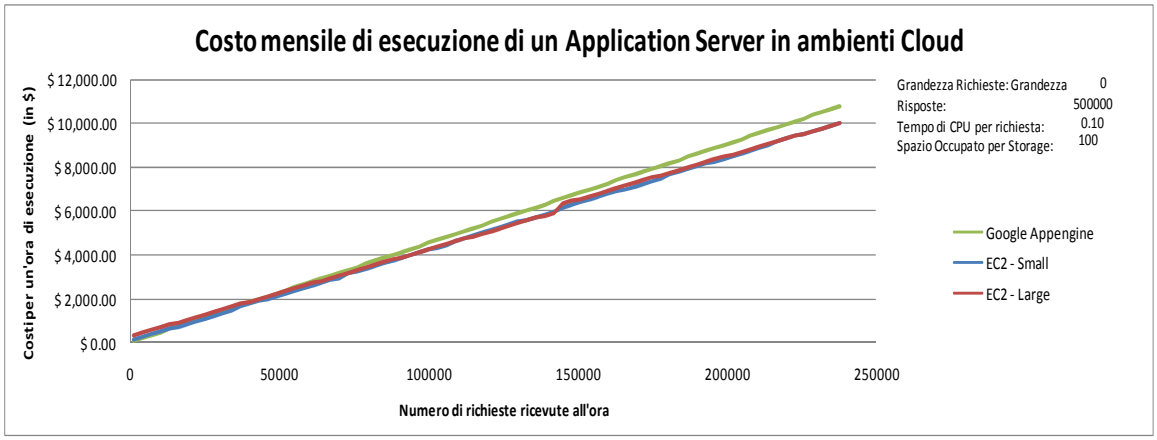
Confronto al variare del traffico generato

Conviene effettuare un confronto delle soluzioni proposte mantenendo il tempo richiesto di CPU costante, ma facendo variare le dimensioni delle richieste e delle risposte, valutando così quanto le tariffe di utilizzo della rete incidano sul prezzo finale. Si vede che se il traffico generato aumenta in maniera considerevole, passando da 5KB per risposta a 50MB, la scelta di implementare il proprio sistema su Amazon EC2 è più conveniente, ma questa scelta è leggermente falsata, in quanto il traffico in ingresso non viene fatturato dal provider IaaS (in promozione fino a Novembre 2010). Come si vede dall'ultimo grafico, dove è stato annullato il traffico in ingresso, le tre diverse soluzioni analizzate sono anche in questo caso molto simili, anche se Google App Engine risulta leggermente meno vantaggioso.


Costo mensile di esecuzione di un Application Server in ambienti Cloud
Costi per un'ora di esecuzione (in $)
$800,00
$700,00
$600,00
$500,00
$400,00
$300,00
$200,00
$100,00
$0,00
0
50000
100000
150000
200000
250000
Numero di richieste ricevute all'ora
Grandezza Richieste: 5000
Grandezza Risposte: 5000
Tempo di CPU per richiesta: 0,1
Spazio Occupato per Storage: 100
Google Appengine
EC2 - Small
EC2 - Large


Costo mensile di esecuzione di un Application Server in ambienti Cloud
Costi per un'ora di esecuzione (in $)
$1.000,00
$900,00
$800,00
$700,00
$600,00
$500,00
$400,00
$300,00
$200,00
$100,00
$0,00
0
50000
100000
150000
200000
250000
Numero di richieste ricevute all'ora
Grandezza Richieste: 10000
Grandezza Risposte: 10000
Tempo di CPU per richiesta: 0,1
Spazio Occupato per Storage: 100
Google Appengine
EC2 - Small
EC2 - Large


Costo mensile di esecuzione di un Application Server in ambienti Cloud
Costi per un'ora di esecuzione (in $)
$25.000,00
$20.000,00
$15.000,00
$10.000,00
$5.000,00
$0,00
0
50000
100000
150000
200000
250000
Numero di richieste ricevute all'ora
Grandezza Richieste: 500000
Grandezza Risposte: 500000
Tempo di CPU per richiesta: 0,1
Spazio Occupato per Storage: 100
Google Appengine
EC2 - Small
EC2 - Large



Commenti

Si vede che economicamente la differenza tra l'utilizzo di una soluzione IaaS o una PaaS per l'esecuzione di una web application sia poco significativa, soprattutto se l'applicazione che si vuole utilizzare non ha dei particolari requisiti di prestazioni o richieste di risorse particolari. Di conseguenza gli elementi che si devono valutare per decidere quale tipologia di provider utilizzare riguardano le funzionalità e le limitazioni che impongono le due diverse offerte: se si verifica che le operazioni da eseguire sono compatibili con una soluzione di tipo Platform-as-a-Service, conviene optare per tale opzione, visto che si possono evitare i problemi di gestione e impostazione che invece si devono affrontare per l'adozione di una soluzione IaaS. Se invece si richiedono particolari funzionalità che non sono presenti su un PaaS, conviene utilizzare delle macchine virtuali personalizzate, di cui si ha il pieno controllo, utilizzando un servizio IaaS, soluzione che comporta però un maggior lavoro per gestire non solo lo sviluppo dell'applicazione, ma anche la gestione della piattaforma di esecuzione.

9. ELASTICDAEMON, AUTOSCALING SU EUCALYPTUS

Eucalyptus è in grado di fornire molti servizi analoghi a quelli di Amazon EC2, tuttavia non sono ancora stati implementati molti dei componenti aggiuntivi che possono risultare molto utili agli utenti, in particolare il servizio di Autoscaling.

È stato implementato un servizio, denominato ElasticDaemon, utilizzabile sulla piattaforma Eucalyptus, che ha le stesse caratteristiche di Autoscaling di Amazon, ossia si consente di definire un insieme di istanze, il cui numero si adatta al carico di lavoro da svolgere. Questa nuova funzionalità è stata progettata per essere compatibile con lo schema di funzionamento di Autoscaling, per poter poi utilizzare un'interfaccia utente molto simile a quella che è stata sviluppata per interagire con i servizi di Amazon.

9.1 Architettura del sistema

Il servizio *ElasticDaemon* è costituito da una serie di componenti che si devono aggiungere ad un'installazione base della versione opensource di Eucalyptus; durante la fase di sviluppo è stata utilizzata la release 1.6.2 di Eucalyptus, è possibile però utilizzare ElasticDaemon anche su versioni differenti della piattaforma. ElasticDaemon è progettato per poter definire un numero variabile di *Autoscaling Cluster*, consentendo quindi ad ogni utente di poter avere uno o più cluster autoscalabili, ognuno indipendente dagli altri. Inoltre è stato aggiunto un sistema di load balancing in grado di interagire con quello di autoscaling, inoltrando il traffico in ingresso in base al numero di istanze attive, aggiornandosi a runtime ogni volta che una macchina virtuale viene avviata o spenta. Il load balancer fornisce anche il supporto alla persistenza delle sessioni a livello di trasporto o applicativo; a questo proposito ElasitcDaemon è in grado di offrire un servizio completamente affidabile per la session stickiness a livello 7, anche in fase di shutdown, cosa che invece non è ancora garantita dai servizi Amazon. Gli elementi necessari al funzionamento del sistema di autoscaling sono più o meno gli stessi che costituiscono la struttura logica dei servizi offerti da Amazon, in particolare sono presenti:

- Un sistema in grado di gestire la *virtualizzazione delle istanze*, fornito dalla piattaforma Eucalyptus, in grado di implementare tutte le funzioni supportate da Amazon EC2, come la

definizione, l'avvio e lo spegnimento di istanze, la gestione delle policy di sicurezza di rete implementate e tutte le altre funzionalità di cui gli utenti necessitano per l'esecuzione delle proprie macchine virtuali. ElasticDaemon interagirà direttamente con il frontend di Eucalyptus per inviare le richieste necessarie per la gestione delle istanze.

- Un sistema di *monitoring* dell'utilizzo di risorse delle istanze, questo servizio deve essere in grado di ottenere le informazioni necessarie da ogni istanza attiva del cluster, memorizzare i dati ed essere in grado di decidere se il cluster è da considerarsi sovraccarico o no. Da questo componente verranno estratte le informazioni necessarie per scegliere quando attivare una nuova istanza o quando spegnerne una già attiva. I servizi che dovrà fornire questo componente devono essere analoghi a quelli offerti da *Amazon CloudWatch*.
- Un *load balancer*: è sicuramente uno dei componenti principali del sistema, deve essere in grado di gestire e inoltrare correttamente tutto il traffico in entrata alle varie istanze attive dell'Autoscaling Cluster. È fondamentale che il load balancer sia in grado di mantenere la persistenza delle sessioni, sia a livello di trasporto, ma soprattutto a livello applicativo, dato che quest'ultima funzionalità è molto spesso richiesta dai sistemi installati su questo tipo di architettura. È necessario inoltre che sia possibile aggiornare la configurazione del load balancer in ogni momento, per poter eseguire correttamente le operazioni di aggiunta e rimozione delle istanze.
- Un *componente di controllo*, che conosce lo stato globale del sistema ed è in grado di comunicare con tutti gli altri elementi, in base alle diverse situazioni che si presentano. Questo componente ha anche il compito di interagire con gli utenti, che potranno interagire con il sistema mediante comandi simili a quelli che si devono utilizzare per l'utilizzo di Amazon Autoscaling.

Tutti i componenti aggiuntivi verranno installati sulla macchina su cui viene eseguito il componente *Cluster Controller* di Eucalyptus, al fine di rendere l'implementazione consistente con l'architettura stessa della piattaforma IaaS installata. Il Cluster Controller è infatti il componente che gestisce il networking tra le istanze in esecuzione sui Node Controller ad esso associati, tutto il traffico proveniente dalla rete esterna infatti dovrà obbligatoriamente passare da questa macchina, visto che è l'unica ad avere un'interfaccia con accesso alla rete pubblica. Il traffico in ingresso viene opportunamente inoltrato alla macchina corretta tramite un sistema di NAT impostato direttamente da Eucalyptus, nel caso in cui a un'istanza venga assegnato un indirizzo ip pubblico. Risulta quindi ragionevole installare il load balancer sullo stesso host del Cluster Controller, visto che in questo modo non si introducono hop aggiuntivi nelle comunicazioni e non si introduce ulteriore overhead; si suppone inoltre che le interfacce di rete installate sul Cluster Controller siano sufficientemente performanti per poter gestire grandi quantità di richieste, in modo da non imporre un limite fisico alle prestazioni dell'intera piattaforma IaaS.

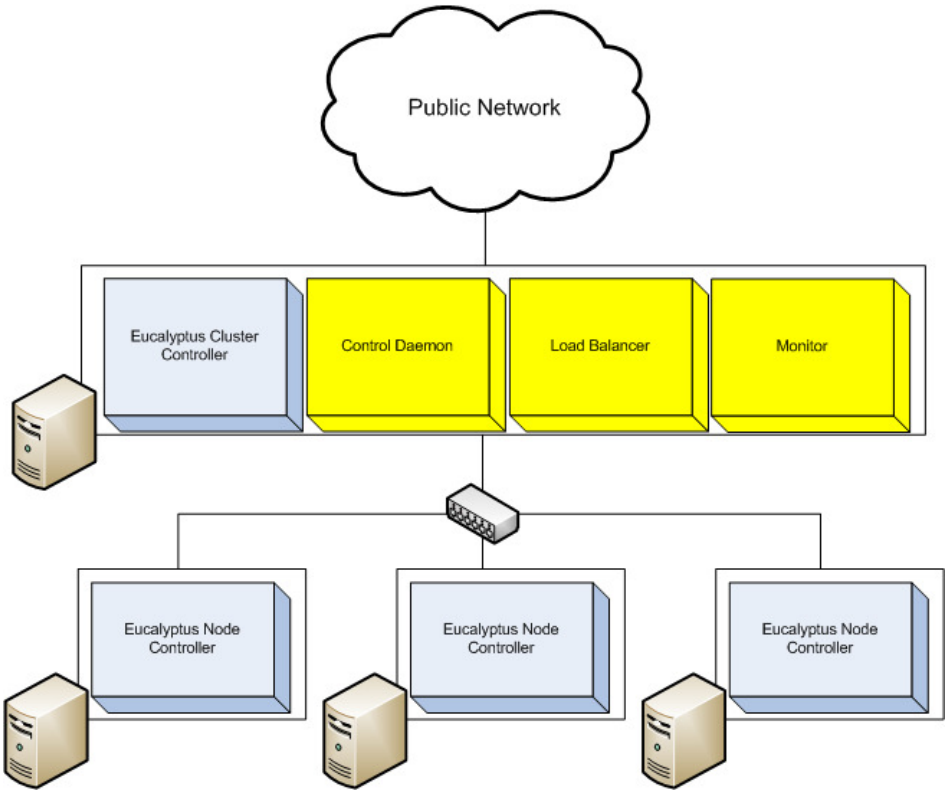


Figura 29 Architettura utilizzata per lo sviluppo di ElasticDaemon

Anche gli altri componenti di ElasticDaemon sono stati installati sulla macchina del Cluster Controller, in questo modo è possibile interagire direttamente con il load balancer ma anche con tutti gli altri componenti di Eucalyptus: una scelta plausibile potrebbe anche essere quella di installare il demone di controllo sulla macchina su cui viene eseguito il frontend di Eucalyptus, ma in questo caso è necessario utilizzare un sistema più complesso per poter effettuare operazioni sui componenti installati sul cluster controller, quale il load balancer, e per monitorare lo stato delle istanze, operazione che risulta invece diretta se eseguita dal cluster controller (il Cluster controller è dotato di un'interfaccia di rete virtuale collegata con la rete privata delle istanze).

9.2 Monitoring delle istanze

Uno dei componenti fondamentali per lo sviluppo di un sistema di autoscaling è il servizio che si occupa di verificare lo stato e l'utilizzo di risorse delle istanze in esecuzione. Il monitor deve innanzitutto verificare che ogni istanza sia attiva e successivamente recuperare le informazioni necessarie per valutarne l'utilizzo di risorse. Come per il servizio *CloudWatch* di Amazon, anche il monitor implementato è in grado di conoscere i valori associati allo stato di utilizzo di diverse risorse, in particolare la percentuale di utilizzo della CPU, la quantità di dati trasmessa in ingresso e in uscita dalla rete e il numero di accessi su disco in lettura e scrittura. Il monitor viene impostato per interrogare le istanze con un certo periodo definito dall'utente, che può essere modificato a runtime.

È necessario definire anche una serie di *metodi statistici* per poter valutare se il cluster di istanze sia sovraccarico o meno, basandosi sui valori medi, massimi, minimi o totali di utilizzo, a seconda di come

l'utente preferisce monitorare le proprie macchine virtuali. Nell'implementazione di ElasticDaemon non è stato inserito il supporto al parametro BreachDuration di Amazon CloudWatch, che potrà essere sviluppato come funzionalità aggiuntiva.

Accesso alle informazioni sull'utilizzo di risorse

Per poter conoscere i parametri necessari a valutare lo stato di utilizzo di determinate risorse è necessario avere accesso diretto alla macchina virtuale oppure poter interrogare l'hypervisor che si occupa della virtualizzazione di questa. Se l'hypervisor utilizzato è KVM, purtroppo non è semplice ottenere informazioni sull'utilizzo di risorse, dato che non è ancora stato sviluppato un sistema in grado di fornire tali dati, in particolare per monitorare la percentuale di utilizzo della CPU. Se si utilizzano tool per il monitoring di virtual machine guest KVM si vede che il valore ritornato non rappresenta l'utilizzo effettivo della CPU virtuale, ma viene rilevata la percentuale di utilizzo della CPU della macchina host, che non costituisce un valore di interesse per il sistema di autoscaling. Per ovviare a questo problema si è deciso di interagire direttamente con la macchina virtuale, recuperando tutti i dati necessari sfruttando le funzionalità offerte dal programma *sysstat*, interrogato mediante un accesso sicuro via ssh. È stato necessario quindi modificare l'immagine da utilizzare su Eucalyptus installando i pacchetti necessari per l'esecuzione di questa applicazione.

Sysstat

Sysstat [62] è un insieme di strumenti di monitoring per sistemi operativi Linux, consente di visualizzare molte informazioni riguardo l'utilizzo attuale delle risorse del sistema; si possono monitorare:

- la percentuale di utilizzo delle CPU
- la quantità di dati trasmessa dalle interfacce di rete
- lo stato della memoria
- la quantità di dati scritti e letti su disco
- dati statistici sulle trasmissioni di tali a livello di rete o di trasporto

La quantità di informazioni che questa suite di strumenti è in grado di fornire è molto grande, basti pensare che solo per il monitoring di una interfaccia di rete si possono consultare più di cento valori. Sfruttando le potenzialità di *sysstat* è stato possibile sviluppare un sistema di monitoring efficiente, che può elaborare una quantità di informazioni molto vasta e precisa. Adeguandosi ai parametri utilizzati da Amazon, il monitor di ElasticDaemon è in grado di monitorare 5 metriche: utilizzo in percentuale di CPU, trasferimenti di dati su disco in scrittura e lettura, quantità di dati trasmessa sull'interfaccia di rete in ingresso e in uscita. Nel caso servisse si possono aggiungere nuove funzionalità per poter monitorare l'utilizzo di risorse mediante metriche diverse, in modo da poter fornire all'utente una gamma più ampia di parametri tra cui scegliere.

Sysstat può essere impostato perché monitori periodicamente i parametri del sistema che l'utente desidera, con un intervallo di tempo personalizzabile, questi parametri però devono essere definiti manualmente in un file di configurazione, diventa quindi impraticabile utilizzare tale tipo di servizio in un ambiente dinamico. La suite di monitoring fornisce una serie di comandi invocabili, ognuno dei quali può essere opportunamente parametrizzato dall'utente, a seconda del tipo di informazioni che desidera ottenere. Uno degli strumenti che si possono utilizzare per monitorare l'utilizzo delle risorse utilizzate da un'istanza è `sar`, che viene invocato con una sintassi del tipo:

```
sar <resource type> <interval> <count>
```

Utilizzando questo strumento è possibile monitorare una determinata risorsa periodicamente, con un intervallo specificato in secondi e per un numero preciso di volte. L'output del programma conterrà tutti i dati relativi alle misurazioni effettuate, seguiti da una riga contenente i valori medi misurati. Per il monitoring delle istanze dell'Autoscaling Cluster una scelta valida può essere di impostare l'intervallo di tempo a 1s e il numero di misurazioni compreso tra 2 e 3 e di utilizzare il valore medio mostrato in output. Il tempo richiesto per effettuare una misurazione è sufficientemente breve (circa 3 secondi) e si evita di ottenere valori non corretti se una delle misure riscontrasse dei valori che si discostano troppo dagli altri, dato che si considera il valor medio dei risultati.

```
bazzu@ubuntu-bazzu:~$ sar -u 1 2
Linux 2.6.32-24-generic (ubuntu-bazzu)  10/13/2010      _i686_  (2 CPU)

10:33:25 AM     CPU     %user     %nice   %system   %iowait    %steal     %idle
10:33:26 AM     all      0.52      0.00      0.52      0.00      0.00     98.96
10:33:27 AM     all      0.49      0.00      1.48      0.00      0.00     98.03
Average:        all      0.51      0.00      1.01      0.00      0.00     98.48
bazzu@ubuntu-bazzu:~$
```

Figura 30 Esempio di output del tool sysstat

Sysstat può monitorare una grande quantità di risorse e di mostrare valori molto precisi di tutte le misurazioni effettuate. Si possono scegliere diversi tipi di misurazioni:

- *Utilizzo di CPU (opzione –u)* in cui vengono misurate le percentuali di utilizzo dei processori, distinti l'uno dall'altro o considerati nel loro insieme. I dati visualizzati mostrano le percentuali in cui ogni tipo di processo (utente, system, ecc) ha occupato il processore.
- *Utilizzo dei dispositivi di I/O (opzione –b)* in cui si misurano la quantità di dati trasferiti sui dispositivi di I/O e il numero di accessi effettuati. Si possono visualizzare il numero di trasferimenti al secondo, da e verso i dischi, oltre alla quantità di dati letti e scritti.
- *Statistiche sul traffico delle interfacce di rete (opzione –n DEV)*, in cui viene analizzato il traffico in ingresso e in uscita dalla interfacce di rete della macchina. I risultati che si ottengono sono la quantità di dati inviata e trasmessa al secondo, il numero di pacchetti inviati e ricevuti, insieme al numero di pacchetti compressi (inviati e ricevuti) e al numero di pacchetti multi cast ricevuti. Per l'analisi del traffico è possibile utilizzare altre opzioni di sar, per ottenere dati a livello IP, come il numero di datagrammi ricevuti, quanti di essi erano frammentati ecc, oppure

sul traffico a livello di trasporto, come il numero di connessioni attivate e terminate al secondo o il numero di pacchetti TCP ricevuti al secondo.

- *Utilizzo di memoria (opzione –r)* in cui `sar` monitora lo stato della memoria della macchina, mostrando la quantità di memoria allocata rispetto a quella libera, insieme ai dati di utilizzo di questa da parte del kernel. È possibile anche visualizzare (utilizzando l'opzione –R) le statistiche relative alla memoria, come il numero di pagine liberate al secondo e il numero di pagine utilizzate per la cache di sistema.

Immagini EMI modificate

Bisogna sottolineare però che per utilizzare gli strumenti sysstat nelle operazioni di monitoring è necessario disporre di immagini modificate, in grado di eseguire gli strumenti di monitoring. Tali immagini devono essere accessibili via ssh dal monitor, perché questi possa raccogliere i dati di cui necessita. Questa complicazione non risulta però così vincolante, infatti basta che il provider fornisca una serie di immagini "ready-to-use" che contengano già gli applicativi richiesti. Anche l'accesso ssh non crea particolari problemi, per mantenere alti i livelli di sicurezza è sufficiente definire un utente che possa eseguire solamente i comandi eseguibili del pacchetto *sysstat*, come `sar`, che sarà collegato all'accesso ssh utilizzato dal monitor.

9.3 Load balancer - HAProxy

Il Load Balancer scelto per l'implementazione del sistema di autoscaling è *HAProxy* [63]. Si tratta di un software open source in grado di gestire una grande quantità di connessioni e di mantenere la persistenza delle sessioni sia a livello di trasporto che a livello applicativo, mediante l'analisi dei cookie HTTP. Il sistema consente un ottimo livello di personalizzazione, fornisce degli strumenti di monitoring e di management per l'amministratore tramite un socket di comunicazione; la versione utilizzata è la *1.5-dev2*, che introduce alcune funzionalità necessarie per la realizzazione del servizio.

Un'istanza di HAProxy è costituita da due componenti principali denominati *frontend* e *backend*: il frontend è l'elemento che si occupa di interagire con l'esterno della rete, è costituito da un listener che ascolta su una determinata coppia IP-porta, che gli utenti contatteranno per accedere al servizio. Ad ogni frontend sono associati uno o più backend, i quali contengono le informazioni relative ai diversi server ai quali dovranno essere inoltrate le richieste ricevute. In ogni elemento di backend sono contenute tutte le informazioni necessarie per inoltrare correttamente ai server le comunicazioni ricevute dal frontend, come indirizzo ip e porta associati ad ognuno di essi. Nel backend sono presenti anche tutte le impostazioni per la gestione dell'eventuale persistenza delle sessioni, come la politica di gestione dei cookie e l'algoritmo da utilizzare per il load balancing delle richieste.

HAProxy è un load balancer software che garantisce ottime prestazioni rispetto a soluzioni analoghe, come evidenziato da alcune analisi prestazionali svolte dal portale *rightscale.com* [56]; oltre a questo è da sottolineare come questo applicativo fornisca quasi tutte le funzionalità richieste per

l'implementazione del sistema di autoscaling proposto; tra le più importanti vi sono il sistema di hot-reconfiguration e la possibilità di impostare i server del backend in modalità *maintenance*.

9.3.1 Funzionalità

Hot reconfiguration

HAProxy può essere riconfigurato senza la necessità di interrompere le connessioni in corso, consentendo di poter aggiungere e togliere server su cui bilanciare il traffico a runtime; questa funzionalità risulta fondamentale per rendere operative le istanze di ogni cluster, dopo che queste sono state avviate per far fronte a un aumento del carico di lavoro. La funzionalità di *soft-restart* si applica semplicemente modificando il file di configurazione di HAProxy e riavviando il load balancer con determinati parametri. Il load balancer si riavvierà facendo in modo di non interrompere le connessioni ancora attive, aggiornando la propria configurazione in base ai nuovi parametri.

Maintenance Mode – Zombie Mode

Il load balancer fornisce inoltre alcune funzioni molto utili per gestire la fase di shutdown delle istanze, quando queste non risultassero più necessarie a fronte di un calo del numero di richieste in arrivo; la procedura di rimozione di un'istanza richiede che tutte le sessioni attive sulla macchina in fase di spegnimento vengano mantenute attive, in modo da rendere queste operazioni del tutto trasparenti all'utente finale. HAProxy consente di impostare un server in modalità *Maintenance*, da qui in avanti detta anche *"Zombie"*: in questa modalità al server non verranno più inoltrate nuove richieste, ma solamente quelle relative alle sessioni ancora attive associate ad esso. È fondamentale impostare il server associato all'istanza in fase di spegnimento in modalità zombie prima di effettuare le operazioni di spegnimento vere e proprie, solo così si può mantenere la persistenza delle sessioni, sia al livello 7 che al livello 4 della pila ISO-OSI.

Per quanto riguarda il mantenimento delle sessioni a livello di trasporto, è sufficiente mettere in modalità "Zombie" il server e attendere il tempo necessario perché una sessione vada in timeout, in questo modo tutte le comunicazioni ancora attive termineranno: potranno terminare correttamente o, nel caso in cui il server avesse dovuto effettuare operazioni troppo lunghe, essere terminate direttamente dal server per timeout.

Bisogna invece porre più attenzione per gestire lo spegnimento di un server mantenendo le sessioni a livello applicativo; il procedimento richiede sempre di attivare la modalità "Zombie" del server preso in considerazione, verificando che tutte le sessioni siano effettivamente terminate. A livello 7 solitamente una sessione è costituita da diverse comunicazioni a livello 4, è necessario quindi che tutte queste richieste relative a una determinata sessione vengano inoltrate al server "Zombie", il quale potrà interagire correttamente con l'utente. Anche le sessioni a livello 7 sono soggette a timeout, tuttavia il timer relativo ad ogni sessione viene riaggiornato ogniqualvolta il server riceve una richiesta appartenente a tale sessione; si deve quindi disporre di un sistema di *monitoring delle sessioni attive* su

ogni server e verificare quando il server che si vuole spegnere non abbia più associata alcuna sessione e solo in questo momento potrà essere spento. È da notare che nel caso peggiore questa situazione potrebbe non accadere mai, se ad esempio un utente continuasse ad inviare richieste relative alla medesima sessione prima che scada il timeout, questa sarà sempre associata al server "Zombie", rendendo impossibile procedere con la fase di spegnimento.

HAProxy fornisce una funzionalità che può essere utilizzata, con opportuni adattamenti, per monitorare le sessioni attive sui diversi server: è possibile definire delle "*sticky-tables*" che mantengono in memoria i dati relativi ad ogni sessione riconosciuta dal load balancer; tuttavia questo sistema risulta poco affidabile per alcune scelte implementative effettuate da parte degli sviluppatori. Per ottenere un servizio di monitoring efficiente è stato necessario introdurre alcune nuove funzionalità al load balancer, basandosi su quelle già presenti per la gestione delle sticky-tables. Le modalità e l'implementazione di queste nuove funzionalità verranno descritte successivamente.

9.3.2 Analisi delle prestazioni di HAProxy

HAProxy è in grado di ottenere buone prestazioni, lo confermano alcune analisi svolte dal portale *Rightscale.com* [56] e dagli sviluppatori stessi del load balancer [64]. Le due analisi che verranno presentate hanno verificato quanto HAProxy sia performante in due scenari completamente diversi: quanto il software è eseguito su un'istanza IaaS e quando invece è eseguito su una macchina fisica.

Richtscale.com – Load balancer su istanze EC2

I test condotti dal team di *rightscale.com* hanno l'obiettivo di verificare quale architettura convenga utilizzare per implementare un cluster di istanze autoscalabile su Amazon EC2, o più in generale su un qualsiasi provider cloud IaaS; una scelta possibile è quella di utilizzare un cluster di istanze gestite dall'autoscaling di Amazon e di avviare un'istanza aggiuntiva sulla quale viene installato un load balancer software (questa soluzione non è altro che la trasposizione dell'architettura che si utilizzerebbe se si dovesse implementare il cluster su host fisici). Un altro tipo di architettura invece è basata sull'utilizzo del servizio di load balancing offerto dal provider, nel caso particolare Amazon Elastic Load Balancing, che va a sostituire l'istanza dedicata al load balancer software.

Una delle metriche valutate è il numero massimo di connessioni al secondo che il sistema può ricevere, oltre ad altri valori quali utilizzo di CPU e di memoria, che però si sono rivelati meno significativi. Tutte le misurazioni sono state effettuate utilizzando un'architettura simile per i diversi load balancer analizzati, in particolare sono stati utilizzati HAProxy, ZeusLB, aiCache e ELB: sono stati avviati più webserver su vari host contenenti una semplice pagina html, che veniva continuamente richiesta da un generatore di carico installato su una macchina esterna (ApacheBench). L'intera struttura è stata installata su istanze EC2, ognuna delle quali aveva installato un webserver (5 istanze); sono state attivate poi istanze aggiuntive per l'esecuzione del load balancer, con l'eccezione di ELB, il quale non richiede di essere installato su un'istanza dedicata. Per la valutazione delle prestazioni di Elastic Load Balancing è stata utilizzata un'architettura leggermente differente, a causa della diversa

implementazione del load balancer stesso; il servizio proposto da Amazon infatti è implementato perché sia in grado di aumentare le risorse utilizzate se il numero di richieste aumentasse. Come generatore di carico è stato utilizzato ApacheBench, installato su una macchina esterna, impostato per inviare 100000 richieste, 100 alla volta al load balancer.

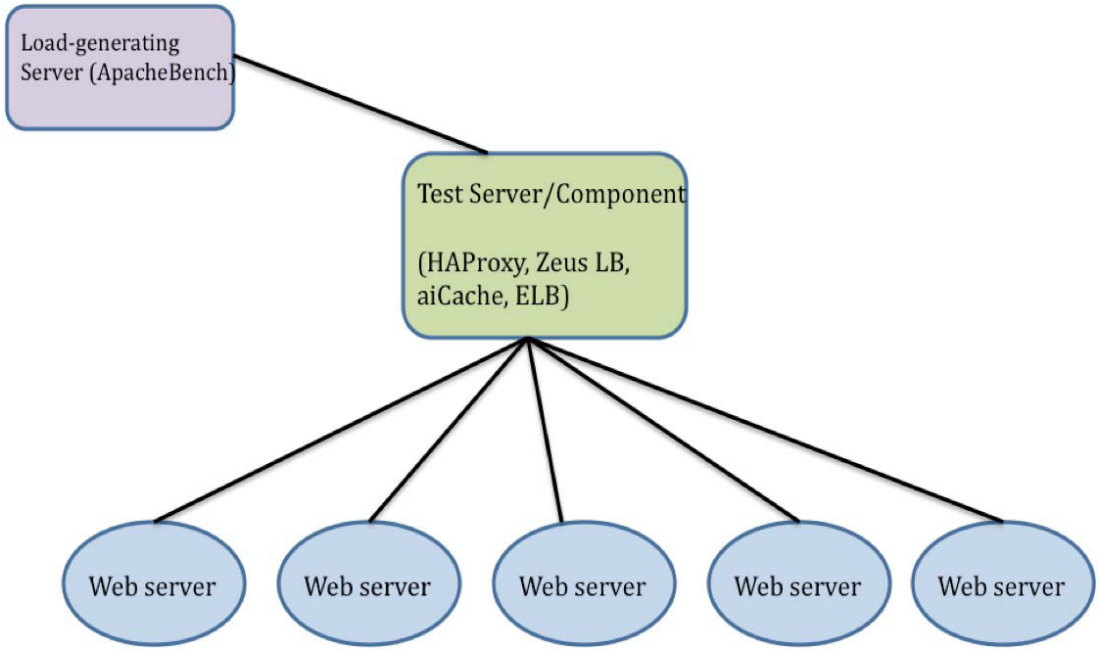


Figura 31 Architettura utilizzata per le misurazioni

I risultati dei test svolti hanno evidenziato che HAProxy e Zeus sono in grado ottenere prestazioni simili quando installati su istanze EC2: HAProxy ha ottenuto un rate di 5239req/s mentre Zeus ha ottenuto un risultato migliore: 6476req/s. Per quanto riguarda ELB, la valutazione è stata effettuata utilizzando un'architettura analoga a quella dei test precedenti, si sono ottenuti risultati non molto buoni, in quanto il numero di connessioni massimo è stato di 2293/sec. Il motivo di questi risultati è da ricercarsi nell'autoscalabilità di ELB: il load balancer viene reso più performante solo se il sistema verifica che le richieste provengono da ip differenti. È stato effettuato un test utilizzando un'architettura più complessa, composta da 25 webserver e 45 generatori di richieste: in questo caso le prestazioni del servizio di Amazon sono aumentate progressivamente, fino a poter rispondere a un numero di richieste superiore a 19.000/sec; queste prestazioni però si sono potute riscontrare solamente dopo un periodo di "scaling-up" di circa 45 minuti, nel quale il servizio si è adattato al carico generato dai tool di misurazione utilizzati. Bisogna sottolineare però che ELB è in grado di ottenere prestazioni così elevate perché il carico viene distribuito su più ip diversi, gestiti tramite un roundrobin DNS associato al nome del load balancer. I risultati quindi non si possono confrontare così facilmente con quelli ottenuti con load balancer installati su singole macchine.

È stata effettuata anche un'analisi su quale possa essere il fattore limitante sulle prestazioni dei load balancer analizzati, si è visto infatti che sia l'utilizzo di CPU, che la banda in entrata non costituiscono un fattore limitante, infatti sono mai stati utilizzati al 100%; a conferma di ciò per il load balancer

HAProxy sono stati eseguiti gli stessi test su un'istanza EC2 di dimensioni massime, ottenendo gli stessi risultati. Uno dei fattori che può limitare il numero di connessioni è stato individuato nel numero di pacchetti al secondo che l'interfaccia di rete virtuale è in grado di ricevere, mediante l'utilizzo del tool ttcp si è visto che le istanze mantenevano un numero di pacchetti al secondo trasmessi di circa 110000. È necessario quindi verificare le capacità dell'interfaccia di rete dell'istanza su cui si vuole implementare il proprio load balancer, ad oggi non è possibile aumentare il numero di packet-per-second sulle interfacce di rete virtuale EC2, se si richiedono prestazioni maggiori rispetto a quelle ottenute si deve optare per una soluzione alternativa, come ELB di Amazon.

Analisi svolte dal team di sviluppo di HAProxy

Sono state eseguite altre misurazioni di performance di HAProxy dal suo stesso team di sviluppo, durante questi test però il load balancer è stato installato su macchine fisiche, dotate di interfacce di rete con prestazioni elevate. Sono state utilizzate tre macchine collegate tra loro, le cui caratteristiche sono riassunte nella tabella seguente:

Machine	Role	Mobo	CPU	Kernel	software
AMD2	Client	ASUS M3A32MVP	AMD Phenom X4/3GHz	2.6.27smp-wt5	inject31
C2D	Proxy	ASUS P5E	intel C2D E8200/2.66GHz	2.6.27smp-wt5	HAProxy-1.3.17-12
AMD1	Server	ASUS M3A32MVP	AMD X2/3.2GHz	2.6.27smp-wt5	HTTPterm 1.3.0

Tabella 12 Macchine utilizzate per effettuare i test di HAProxy

Un generatore di richieste HTTP è stato installato sulla macchina *am2* il load balancer su c2d e un web server su am1, le connessioni tra una macchina e l'altra sono dirette, nessuno switch intermedio è stato utilizzato. Il load balancer è stato configurato per inoltrare tutte le richieste in arrivo al web server, che si occuperà di generare le risposte. Durante il test uno script interagisce con il load generator avviando delle richieste per oggetti presenti sul web server di dimensioni sempre più grandi, da 64kB a 10MB, vengono generate da 5000 a 1000 richieste contemporanee per intervalli di un minuto per ogni oggetto. I valori ottenuti sono quelli riportati nel grafico in Figura 32, in cui i valori in verde rappresentano il numero di richieste soddisfatte al secondo, mentre in verde è riportata la banda utilizzata dalle trasmissioni.

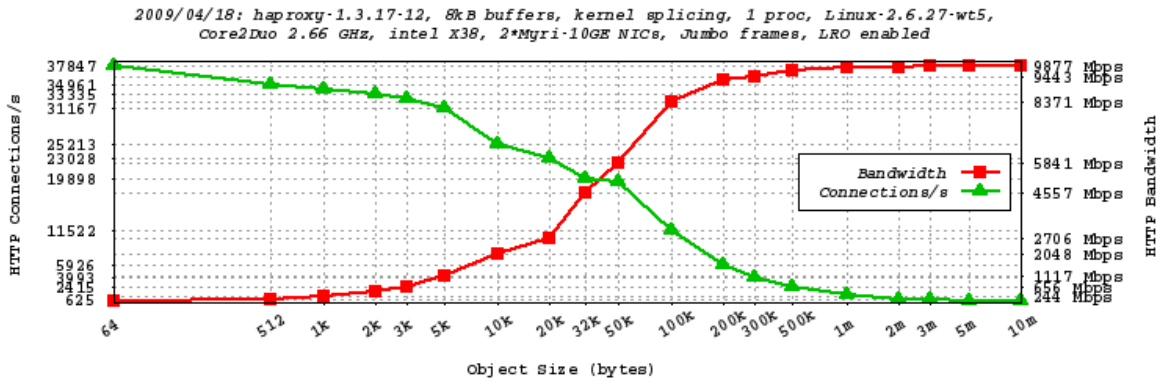


Figura 32 Grafico dei risultati dei test svolti dal team di sviluppo di HAProxy

Si vede come il load balancer riesca a gestire un numero molto elevato di connessioni, arrivando a saturare la banda disponibile dell'interfaccia di rete. Il numero di connessioni tende a diminuire con l'aumento della grandezza degli oggetti, a fronte di un aumento della banda utilizzata. Anche in questi test l'utilizzo di CPU non è stato un fattore limitante alle prestazioni del load balancer, infatti in tutte le misurazioni la percentuale di tempo di CPU utilizzato non ha mai raggiunti livelli superiori al 20%.

```
root@c2d:tmp# vmstat 1
       procs                      memory      swap          io     system       CPU
       r  b  w   swpd   free   buff  cache   si   so    bi    bo    in    cs  us sy id
       1  0  0      0 1951376   3820  19868    0    0     0     0 25729 23965  1 15 84
       0  0  0      0 1950652   3820  19868    0    0     0     0 25744 23818  3 17 80
       0  0  0      0 1950632   3820  19868    0    0     0     0 25720 24652  1 18 80
       0  0  0      0 1949512   3820  19868    0    0     0     0 25531 24047  3 16 81
       1  0  0      0 1948484   3820  19868    0    0     0     0 25911 22706  2 19 79
       0  0  0      0 1949388   3820  19868    0    0     0     0 26189 23757  3 15 82
       1  0  0      0 1948460   3820  19868    0    0     0     0 25811 23766  1 20 79
```

Commenti

Dai risultati ottenuti nei test descritti è evidente che HAProxy risulta molto più performante se installato su una macchina fisica, dato che i limiti imposti dalla virtualizzazione ne diminuiscono di molto le prestazioni. Si è visto inoltre che il software non incide drasticamente sull'utilizzo della CPU e della memoria, che non costituiscono un fattore limitante, se non in minima parte.

Questi test confermano come la scelta di implementare il load balancer sulla macchina del *Cluster Controller* di Eucalyptus sia una scelta coerente, visto che in questo modo sarà possibile sfruttare al massimo le potenzialità di HAProxy, senza incidere negativamente sulle prestazioni del Cluster Controller, visto che HAProxy non richiede una quantità eccessiva di risorse di calcolo. Inoltre implementare il load balancer al di fuori delle istanze rende l'intero sistema logicamente simile ad ELB, consentendo quindi di poter implementare servizi totalmente analoghi a quelli offerti da Amazon.

9.3.3 Configurazione di HAProxy

HAProxy viene configurato tramite un file di testo da fornire in input al momento dell'esecuzione, in esso sono contenute tutte le informazioni necessarie per il bilanciamento delle connessioni in arrivo. Si devono configurare i due elementi principali per ogni cluster di istanze: il *frontend* e il *backend*.

Frontend

Le opzioni che devono essere definite nel file di configurazione sono:

- *mode:* definisce la modalità con cui sarà eseguito il load balancing su questo frontend, i valori possono essere:
    - *TCP*: il load balancer lavorerà in TCP mode, ossia verranno inoltrare alla stessa istanza tutti i pacchetti di uno stesso flusso tcp, mentre non sono considerate tutte le informazioni fornite dal livello applicativo sovrastante. Questa modalità rende il load balancer in grado di mantenere la persistenza delle sessioni al layer 4, si adatta per flussi SSL, SSH, SMTP, ecc. È la modalità più semplice da gestire, tuttavia non si

adatta a tutte quelle applicazioni che necessitano di mantenere la persistenza delle sessioni a livello applicativo, come le applicazioni web.

  - *HTTP*: in questa modalità, oltre ad essere mantenute tutte le sessioni a livello di trasporto, tutte le richieste di tipo HTTP pervenute al frontend verranno analizzate dal load balancer, successivamente verrà deciso su quale server del backend dovrà essere inoltrata la richiesta. In questa modalità si può ottenere la persistenza delle sessioni a livello 7, oltre a poter applicare filtri e condizioni basati sulle informazioni contenute nei pacchetti HTTP. Il fatto di poter mantenere le sessioni "sticky", consente di utilizzare questa modalità per bilanciare il carico di richieste di un'applicazione web in maniera totalmente trasparente all'utente finale; in questa modalità quando una sessione viene inoltrata su un determinato server, tutte le successive richieste appartenenti alla stessa sessione saranno inviate al medesimo server, che conosce tutte le informazioni necessarie per interagire correttamente con l'utente.
  - *Health*:in questa modalità il frontend risponderà solamente "OK" alle connessioni in arrivo, chiudendo poi la comunicazione; viene utilizzata solamente per rispondere a sistemi esterni di health check.

- *Bind ip:port* definisce uno o più indirizzi (e relative porte) su cui ascolterà il frontend; si possono inoltre definire ulteriori parametri quali l'interfaccia fisica di rete e altre impostazioni per la gestione delle statistiche. Si può definire anche il parametro "*transparent*" che interagisce automaticamente con il sistema linux associando all'interfaccia di rete gli indirizzi definiti, avviando di fatto l'ascolto su tali IP se questi non fossero già definiti nelle impostazioni di rete del sistema. Il supporto all'opzione transparent è attivo solo per alcune configurazioni particolari del sistema proposto.
- *use_backend* <*nome*> *if* <*condizione*> le connessioni in entrata saranno inoltrate al backend definito in questa istruzione se la condizione è verificata. È possibile definire più backend con diverse condizioni associate. Nell'implementazione del sistema di autoscaling questa impostazione non risulta particolarmente utile, dato che verrà utilizzato un solo backend per ogni frontend.
- *default_backend* <*nome*> definisce il backend da utilizzare per il bilanciamento delle connessioni in entrata quando nessuna delle condizioni di use_backend è verificata o se non ve ne sono presenti.

Backend

I parametri che vanno forniti per ogni backend sono:

- *option HTTPclose*: il load balancer analizza l'header HTTP e verifica se sia presente un "Connection close" su entrambi i lati della comunicazione quando è necessario, se così non

fosse, aggiunge l'header dal lato mancante, consentendo la chiusura corretta della connessione. Questa opzione risulta importante per ottenere la persistenza delle sessioni a livello 7.

- *option forwardfor*: abilita l'inserimento dell'header HTTP X-forwarded-for su tutti i pacchetti inoltrati dal frontend ai server dei backend; questa opzione è utile in quanto i server backend vedono come indirizzo sorgente delle richieste quello di HAProxy e non quello del client. Abilitando l'opzione forwardfor i server di backend potranno essere consapevoli che la comunicazione ha come intermediario un load balancer. È utile anche utilizzare questa opzione quando si vuole avere stickiness a livello 7.
- *cookie <nome> <opzioni>* : è l'opzione fondamentale per gestire la persistenza delle sessioni a livello 7. Si deve fornire il nome del cookie da utilizzare e delle opzioni per scegliere in che modo il load balancer dovrà implementare la session stickiness. Queste metodologie verranno spiegate successivamente in una sezione dedicata.
- *Stick-table type <tipo> size <size> expire <time>* : questa opzione attiva una tabella in cui viene salvata una entry per ognuna delle sessioni sticky a livello 7 attive. Si deve definire innanzitutto il tipo di dati salvati (di tipo ip, int o string), la dimensione massima e un intervallo di tempo dopo il quale l'entry della tabella si considera scaduta. A questa istruzione si associa *stick store-request <pattern>* che definisce quale tipo di informazione dovrà essere inserita della tabella, possibili scelte sono: l'indirizzo IP sorgente, l'indirizzo ip di destinazione o la porta sorgente o quella di destinazione. Bisogna sottolineare però che le informazioni in questa tabella vengono sovrascritte se i dati da inserire coincidono con quelli di una entry già presente; ad esempio se da uno stesso indirizzo IP si collegassero due utenti, ognuno assegnato a un backend server differente, ad ogni comunicazione dell'uno, l'entry associata all'altro verrà sovrascritta, rendendo quindi poco chiara l'informazione si quante effettive sessioni sono attualmente attive sul cluster. Verrà descritto in seguito come sia stato risolto questo problema per poter utilizzare le informazioni della *stick-table* nel sistema di autoscaling implementato.
- *Server <ip>:<porta> cookie <nome>* : si deve ripetere questa istruzione per ciascun server associato al backend in questione, si fornendo l'indirizzo IP (o il nome) del server, oltre che alla porta su cui esso dovrà essere contattato. Non è necessario che la porta definita nel frontend coincida con quelle definite nel backend, si possono anche definire porte diverse per server diversi se necessario. Se si richiede che le sessioni a livello 7 siano persistenti, si deve definire anche un nome univoco del server (rispetto agli altri definiti nel cluster) che verrà inserito come identificativo nei cookie aggiuntivi dal load balancer.

9.3.4 Session stickiness a livello applicativo

Le istruzioni per mantenere la persistenza delle sessioni a livello 7 si devono inserire all'interno della sezione relativa al componente backend. HAProxy implementa tre diversi metodi per mantenere le sessioni, tutti basati sull'elaborazione dei cookie contenuti nelle richieste HTTP ricevute.

- *appsession <nome>* – In questa modalità si deve specificare il nome di un cookie utilizzato dall'applicazione implementata sui server del backend (ad esempio per i server Tomcat il cookie può essere JSESSIONID), HAProxy analizzerà tutte le comunicazioni HTTP e verificherà la presenza di un cookie con quel nome; la prima parte del contenuto di questo verrà analizzata, fino ad un massimo di 64 caratteri, e utilizzata per decidere se si tratti di una nuova sessione o di una già conosciuta. Nel caso il load balancer non riconoscesse il contenuto del cookie, procederà ad aggiungere una entry in una tabella hash nella quale sono contenuti i dati relativi al contenuto del cookie e al server a cui è associata la sessione. Questo metodo è del tutto trasparente all'utente finale, tuttavia richiede una conoscenza specifica dell'applicazione in esecuzione sui server di backend dato che si deve fornire il nome del cookie di sessione utilizzato. Dato che il sistema analizza solo i primi 64 caratteri del contenuto del cookie, che solitamente contiene l'ID della sessione, può accadere, anche se raramente, che due cookie di sessione condividano lo stesso prefisso; tale situazione non risulta così problematica, infatti il load balancer considererà i messaggi ricevuti (appartenenti alle due sessioni diverse) come se appartenessero ad un'unica sessione, di conseguenza li inoltrerà allo stesso application server, garantendo comunque la stickiness delle sessioni. È evidente che la scelta del cookie da analizzare deve essere effettuata in maniera opportuna, in modo che a sessioni diverse corrispondano, almeno per la maggior parte dei casi, cookie con contenuto diverso.
- *Cookie <nome> prefix* – Il load balancer anche in questo caso verifica la presenza di un determinato cookie, che deve quindi essere associato all'applicazione in esecuzione sui server. In questa modalità però il server non manterrà alcuna informazione sulla sessione e di come gestirne la persistenza, perché tutte queste informazioni vengono aggiunte al cookie preso in esame. Quando il load balancer intercetta un cookie con il nome corretto, ne modifica il contenuto inviando all'utente un cookie con lo stesso nome ma con alcuni dati aggiunti come *prefisso* del contenuto originario; l'utente quindi visualizzerà un cookie della forma: nome=JSESSIONID contenuto=serv2~Cnoas46Cbn…. Dove il prefisso *serv2~* è un flag necessario al load balancer per il corretto smistamento delle richieste future. Quando l'utente esterno invierà una richiesta all'applicazione HAProxy, dopo aver analizzato il cookie e scelto il backend-server appropriato, rimuoverà il prefisso del cookie prima di inviarlo all'istanza, rendendo di conseguenza questo passaggio trasparente dal punto di vista dell'application server. A livello utente invece il prefisso sul contenuto del cookie rimane visibile.
- *Cookie <nome> insert* – In questa modalità il load balancer ogniqualvolta riceve una richiesta in cui non compare un cookie con il nome definito, oppure questo compare in un formato diverso da quello previsto, inserisce un nuovo cookie con il nome scelto dall'utente e con il contenuto necessario per le future elaborazioni. In questo caso l'amministratore deve verificare che il nome del cookie scelto non corrisponda con altri utilizzati dall'applicazione, in quanto

HAProxy effettuerà una completa sostituzione del contenuto in caso di conflitto. Anche in questo caso a livello utente si nota la presenza di un cookie aggiuntivo.

La seconda e la terza soluzione sono sicuramente le più scalabili e le più performanti per lo scenario di utilizzo proposto, infatti si presume che il numero di richieste dirette al cluster di istanze sia molto elevato e proveniente da molti utenti diversi; la soluzione di salvare in memoria dati relativi ad ogni sessione potrebbe essere poco performante in questa situazione, è sicuramente più conveniente inserire le informazioni sui cookie e accedervi solo quando necessario. Inoltre i dati contenuti nella tabella hash vengono persi dopo un reset del load balancer, rendendo di fatto inutilizzabile questa modalità nell'implementazione del servizio di autoscaling.

Anche il servizio di load balancing offerto da Amazon, *Elastic Load Balancer*, utilizza una modalità simile a *cookie insert*: viene aggiunto un cookie speciale ad ogni nuova sessione. Ci sono due modalità di gestione di questo cookie: una più semplice che verifica solamente quando sia necessario inserire un nuovo cookie all'avvio di una sessione, l'altra analizza anche il seguente traffico HTTP e controlla quando l'applicazione invia al client il segnale di cancellazione del cookie. Quando ciò avviene il load balancer di Amazon cancellerà anche il proprio cookie aggiuntivo, concludendo completamente la sessione anche per il bilanciamento del traffico.

9.3.5 Esempio di File di Configurazione

Un file di configurazione completo per la configurazione del load balancer deve contenere anche le sezioni global e default, dove si possono definire alcuni parametri generali, che verranno editati da tutti gli altri elementi definiti nelle istruzioni successive.

```
global
    log 127.0.0.1  local0 debug
    stats socket socket_Cluster1 level admin

defaults
    log global
    mode    HTTP
    maxconn    2000
    option persist
    option redispatch
    #algoritmo di bilanciamento
    maxconn 2000
    contimeout      5000
    clitimeout      50000
    srvtimeout      50000
    stats enable
    balance roundrobin

frontend Cluster1
  log global
    mode HTTP
    bind 192.168.20.254:5001
    default_backend Cluster1_backend

backend Cluster1_backend
    log global
```

```
    option HTTPclose
    option forwardfor
    stick-table type ip size 1m expire 1m
    stick store-request src
    cookie JSESSIONID prefix nocache
    #Cluster1 backend servers:
    server i-233243 192.168.20.20:8080 cookie i-233243
    server i-546782 192.168.20.21:8080 cookie i-546782
```

9.3.6 Stick Table

HAProxy mantiene in memoria i dati relativi alle sessioni attive a livello 7 solamente se è attiva l'opzione *appsession*, mentre quando si utilizzano le opzioni cookie *insert* o *persist* nessun dato viene salvato, dato che tutte le informazioni necessarie al routing corretto delle richieste è contenuto nei cookie modificati. Viene fornita comunque una funzione che permette di monitorare le sessioni attive per ogni backend in una tabella detta *sticky-table*. I dati contenuti in questa tabella non sono fondamentali per il mantenimento delle sessioni a livello 7, servono solo come sistema di monitoring e statistico, per poter verificare in ogni momento lo stato attuale delle sessioni attive sugli application server su cui viene bilanciato il traffico in arrivo. Si possono memorizzare non solo i dati relativi al client e l'id del server a cui è associata la sessione, ma anche alcune informazioni riguardo lo stato e l'utilizzo delle connessioni attive, oltre ad alcune funzioni statistiche sui parametri delle richieste HTTP ricevute.

I dati relativi a una sessione vengono aggiornati ogni volta che una richiesta ad essa associata viene analizzata dal load balancer, in particolare viene riaggiornato l'*expire-time*, fondamentale per il monitoraggio dello stato delle sessioni associato a un server.

Problematiche

Il servizio di *stick table* fornito da HAProxy non risulta però essere completamente affidabile per essere utilizzato nell'implementazione del nostro sistema di autoscaling, per due motivi principali:

- le dimensioni della tabella sono *finite*, il sistema quando verifica che la tabella è piena, scarterà automaticamente l'entry più vecchia, tralasciandone le informazioni. Dato che questo evento è facile che si verifichi, soprattutto se si sta monitorando un backend composto da molti server, non è certo che si possa avere una visione completa dello stato del sistema.
  I problemi maggiori si possono verificare durante la procedura di shutdown di un server: in questa situazione è importante poter monitorare tutte le sessioni attive su tal server, tuttavia a causa di un overflow della tabella, i dati delle sessioni associate al server possono essere scartati dal load balancer per liberare spazio a nuove sessioni associate ai server ancora attivi, in questo modo non si potrà avere la certezza che tutte le sessioni associate al server in fase di spegnimento siano effettivamente scadute.
- Le chiavi associate ad ogni entry di una tabella *non sono univoche* per ogni sessione, ma sono costituite da parametri caratteristici della connessione di livello più basso, come indirizzo IP sorgente o di destinazione, porta di destinazione o parametri estratti dall'header HTTP. Può

succedere quindi che due sessioni che condividono lo stesso parametro utilizzato per l'indicizzazione della tabella *sovrascrivano* una l'entry dell'altra. Un caso in cui questo può avvenire si presenta quando due client, collegati alla rete pubblica tramite NAT, avviano due sessioni sul load balancer. L'indirizzo IP sorgente delle due sessioni sarà lo stesso, tuttavia il load balancer potrebbe inoltrare le richieste di un client su un server del backend, mentre quelle dell'altro su un server differente. L'entry della sticky table associata all'IP dei due client sarà unica, e verrà aggiornata ad ogni richiesta inviata da ognuno dei due client, cancellando le informazioni relative alla sessione dell'altro server. Può succedere quindi che monitorando lo stato di un server si verifichi erroneamente che non sono più attive sessioni su di esso, quando invece queste sono state semplicemente sostituite nella sticky table da entry la cui chiave è la stessa.

Soluzioni adottate

Per risolvere le problematiche elencate è stato necessario apportare alcune modifiche al codice sorgente di HAProxy, in particolare nella gestione delle sticky table. La soluzione implementata modifica la politica di gestione delle stick table, in modo che vengano aggiunte entry solamente relative a sessioni associate ai server in modalità "*zombie*". Questa soluzione risulta perfettamente utilizzabile per il monitoraggio delle sessioni durante lo shutdown di un server, infatti non risultano problematiche né la finitezza delle dimensioni della tabella, né la non univocità delle chiavi. Non è importante ora se una entry viene *scartata per un overflow della tabella*, dato che è sufficiente poter conoscere se sono attive delle sessioni e non la loro quantità precisa; per questo motivo non è necessario nemmeno gestire il caso di una *"sovrapposizione" di sue sessioni*, dato che basta sapere quale è l'ultima di esse ad essere stata attiva (ossia l'ultima che ha sovrascritto l'entry ad essa associata).

La soluzione descritta risulta funzionante solamente se un solo singolo server si trova in modalità zombie, i problemi descritti precedentemente si ripresenterebbero qualora due server di uno stesso backend venissero impostati in "zombie" mode. Questa limitazione non è da considerarsi problematica, infatti in un sistema elastico si presume che in fase di scaling-down venga sempre spenta un'istanza alla volta, al fine di evitare fenomeni di *underprovisioning*.

9.3.7 Hot-reconfiguration

Per effettuare la procedure di *hot-reconfiguration* è sufficiente avviare il load balancer con un flag aggiuntivo e fornire il'PID (process-ID) del processo di HAProxy da "aggiornare"; quando riceve questo tipo di segnale HAProxy aggiornerà la propria configurazione in base al file passato come parametro, ma senza interrompere le connessioni attive. Il load balancer dopo aver effettuato una hot-reconfiguration perde alcuni dati che manteneva in memoria, come i dati statistici, le informazioni salvate nelle sticky-table e i dati memorizzati per gestire la stickiness mediante l'opzione appsession. Nello sviluppo del nuovo servizio per Eucalyptus i dati statistici non risultano importanti, quindi non costituiscono un problema, come del resto la perdita di dati delle sticky-table, infatti il riavvio del load

balancer avviene sempre per aggiungere o rimuovere un server dal backend, situazioni in cui perdere i dati della sticky table non provoca danni.

La funzionalità di hot-reconfiguration sarà utilizzata dal servizio di autoscaling per modificare e aggiungere i diversi server di backend, associati alle istanze attive in quel determinato momento. Si deve però verificare che le operazioni di restart avvengano in istanti in cui non sia problematica la perdita dei dati in memoria del load balancer.

9.3.8 Socket Unix

HAProxy consente all'amministratore di accedere a un *socket* dal quale è possibile ricevere informazioni sullo stato del load balancer e alcune statistiche sul traffico in entrata e in uscita; inoltre tramite questo socket si possono inviare alcuni comandi come ad esempio "*enable server*" e "*disable server*", che rendono rispettivamente un server attivo e in stato "zombie". Le comunicazioni di comandi attraverso questo socket sono state utilizzate per impostare la modalità zombie di un server e per verificare la presenza di sessioni attive su di esso. Questo punto di accesso si è rivelato di grande utilità, soprattutto perché può ricevere molti tipi di comandi, alcuni già implementati dal team di sviluppo come quelli già citati, ma ha reso possibile implementare nuove funzionalità modificando il codice che si occupa della gestione delle comunicazioni in ingresso e uscita su questo socket.

9.3.9 HaTop

Per monitorare lo stato del load balancer durante la fase di sviluppo è stato utilizzato *HaTop* [65], un programma che simula le funzioni del comando top di linux, ma per il monitoring del load balancer HAProxy. Questo strumento interagisce con il socket unix di comunicazione reso disponibile da HAProxy, mostrando a schermo i risultati in forma tabellare. Sono presenti 4 tipi di visualizzazioni:

- *STATUS*: per visualizzare lo stato del load balancer e dei suoi componenti: si può visualizzare lo stato attuale di ogni server e il numero di sessioni attive su di esso
- *TRAFFIC*: per monitorare i dati statistici sul traffico di dati sui diversi server
- *HTTP*: fornisce statistiche sul traffico HTTP
- *ERRORS*: mantiene tutti i contatori degli errori rilevati
- *CLI*: simula una sorta di shell interna, che consente all'amministratore di interagire direttamente con il load balancer, inviando comandi per ricevere statistiche o per inviare ai server comandi quali enable/disable server o get/set weight o altri . Questa interfaccia è risultata molto utile durante la fase di testing delle funzionalità aggiunte modificando il codice sorgente.

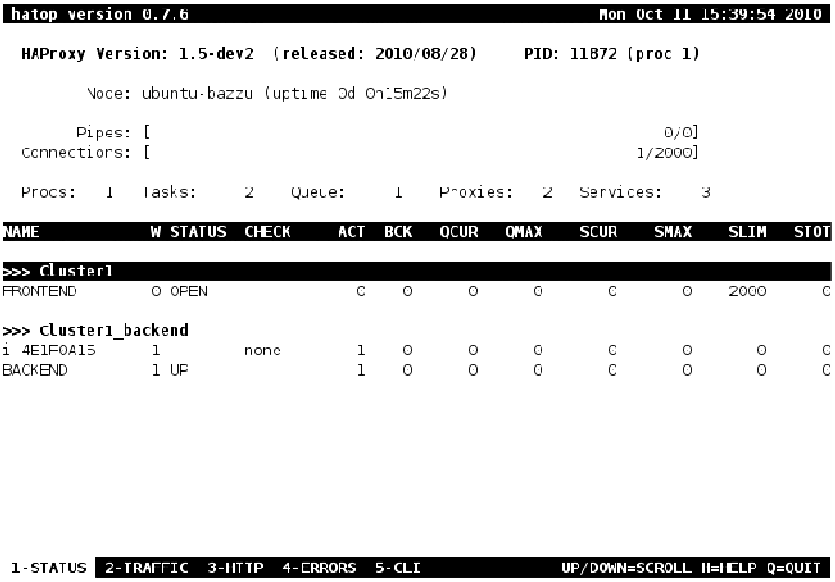


Figura 33 Screenshot dell'interfaccia di HaTop

9.3.10 Implementazione e Utilizzo

Per installare il load balancer è sufficiente disporre del file eseguibile e del file di configurazione da passare come parametro durante l'avvio. È possibile avviare HAProxy in modalità standard o in modalità demone, che risulta più versatile della precedente, soprattutto in ambiente server. Per avviare il load balancer viene utilizzato un comando del tipo: *`./HAProxy -D -f config_file -sf [pid]:`* il flag -D informa il load balancer di eseguire in modalità demone, -f nomefile indica il file di configurazione che si deve utilizzare. Il flag -sf indica che si dovrà effettuare un *soft-restart* del processo il cui ID è specificato, in questo modo è possibile aggiornare la configurazione del load balancer, aggiungendo e rimuovendo server o modificando altri parametri, mantenendo tutte le funzionalità di stickiness delle sessioni e evitando di interrompere eventuali connessioni ancora attive. Utilizzare questo tipo di riavvio in fase di runtime è totalmente trasparente all'utente finale, infatti nessun tipo di connessione viene disturbata da questa operazione, né quelle a livello di trasporto, né le sessioni a livello applicativo.

Per poter gestire in maniera efficiente la presenza di più cluster di macchine si avvierà un'istanza dedicata di HAProxy per ognuno di essi, gestibile in maniera indipendente rispetto alle altre. In questo modo si ottiene anche un buon livello di fault tolerance, in quanto se uno dei load balancer dovesse terminare improvvisamente, solo uno dei cluster attivi diventerà irraggiungibile. Se invece si dovesse utilizzare un'unica istanza del load balancer per tutti i cluster, una terminazione improvvisa sarebbe decisamente più problematica.

9.3.11 Struttura del codice sorgente

Per poter implementare alcune funzioni aggiuntive è stato analizzato nel dettaglio parte del codice sorgente di HAProxy versione 1.5-dev2. Il programma è scritto interamente in linguaggio C ed è dotato di una discreta descrizione tramite commenti. Quasi tutti i file .c sono associati a due file .h con lo

stesso nome, di cui uno definisce i tipi di dato e le strutture utilizzate e l'altro le funzioni pubbliche implementate. Ogni elemento del load balancer è rappresentato da una struttura dati particolare, che a sua volta può contenere altre strutture al suo interno; quasi tutte le strutture principali contengono un puntatore *next* a una struttura dello stesso tipo, creando una sorta di lista concatenata di tutti gli elementi. Essendo tutti gli elementi in memoria collegati tra loro tramite puntatori, risulta abbastanza complesso apportare modifiche al codice, è necessario infatti verificare che tutti i cambiamenti apportati mantengano lo stato del sistema consistente, in particolare per evitare accessi in memoria non corretti.

Le strutture più importanti definite nel codice sorgente sono *proxy e server*: il primo rappresenta o un frontend o un backend, in questa struttura sono memorizzati tutti i parametri generali associati a tali elementi; ad ogni struttura *proxy* è associata una lista di strutture *server*, che contengono tutte le informazioni e i parametri dei diversi server associati al *proxy* in questione. Altre strutture utilizzate sono *session* e *stick_table*.

9.3.12 Modifiche al codice sorgente

Per poter sfruttare tutte le potenzialità di HAProxy è stato necessario modificarne il codice sorgente in alcuni punti, in modo da aggiungere alcune funzionalità necessarie per l'utilizzo in collaborazione con gli altri elementi del sistema autoscalabile. Molte funzioni sono state implementate per essere utilizzate tramite il socket unix fornito dal load balancer. In particolare è stato implementato un filtro per l'opzione di dump delle tabelle per permettere al sistema di verificare in maniera molto veloce se sono presenti delle sessioni attive su un determinato server. HAProxy fornisce dalla versione 1.5 un comando, utilizzabile tramite il socket unix, per visualizzare il contenuto di una sticky table, fornendo il nome del backend a cui essa è associata. I dati visualizzabili sono molteplici, a seconda di come è stata definita la sticky table, ma il dato di fondamentale interesse per la nostra applicazione è il valore server_id; questo parametro rappresenta l'ID del server a cui è associata una determinata sessione. È stato necessario però applicare a questi risultati un filtro per ottenere le informazioni relative alle sessioni attive su un server a partire dal nome e non dall'ID, visto che questi vengono scelti da HAProxy in maniera automatica, mentre i nomi dei server sono personalizzabili. Questa funzionalità però non si limita a filtrare il contenuto della tabella, infatti l'informazione di cui si necessita per il corretto funzionamento dell'applicazione è quella di sapere se ci sono sessioni attive su un determinato server o meno. La procedura implementata effettua una scansione del contenuto della tabella, cercando una corrispondenza tra l'ID associato al nome del server passato come parametro e l'id di ogni entry; quando una corrispondenza viene riscontrata, la procedura invia un messaggio di output e termina la sia esecuzione, evitando di continuare la scansione dell'intera tabella, operazione che non aggiunge informazioni utili di alcun tipo. Il servizio implementato si può richiamare tramite una comunicazione sul socket unix del tipo: `show table <nome> data.act_session srv <srvname>`, dove il nome si riferisce al backend di cui si vuole visualizzare la tabella e srvname è il nome del server preso in considerazione.

Le modifiche apportate al codice sorgente riguardano le funzioni che gestiscono l'interazione con i dati ricevuti sul socket unix, il dumping dei dati delle sticky-table e le operazioni di filtraggio delle entry ottenute.

Filtro act_sessions srv <srvname>

Per prima cosa sono stati definiti i tipi di dati mancanti per poter applicare un filtro personalizzato sul nome dei server, passato come array di char. Nel file *types/stick_table.h* si è definito il tipo di dati STD_T_STRING aggiungendo l'istruzione:

```
enum {
        STD_T_SINT = 0,             /* data is of type signed int */
        STD_T_UINT,                 /* data is of type unsigned int */
        STD_T_ULL,                  /* data is of type unsigned long long */
        STD_T_FRQP,                 /* data is of type freq_ctr_period */
        STD_T_STRING,                    /* data is of type string */
};
```

Per definire il tipo di dato "act_session", necessario poi per applicare il filtro, si deve aggiungere nel file *stick_table.c* (che importa il file stick_table.h precedente) la definizione di tale dato:

```
struct stktable_data_type stktable_data_types[STKTABLE_DATA_TYPES] = {
        [STKTABLE_DT_SERVER_ID]     = { .name = "server_id",       .std_type =
STD_T_SINT  },
…...
```

Infine nel file *standard.h* è stato aggiungo un comparatore srv modificando la sezione di codice:

```
/* operators to compare values. They're ordered that way so that the lowest bit
 * serves as a negation for the test and contains all tests that are not equal.
 */
enum {
        STD_OP_LE = 0, STD_OP_GT = 1,
        STD_OP_EQ = 2, STD_OP_NE = 3,
        STD_OP_GE = 4, STD_OP_LT = 5,
        STD_OP_SRV = 5,
};
```

Nel file *standard.c* associato si deve verificare la presenza della keywork "srv" utilizzata come comparatore:

```
}else if (*str == 's' && str[1] == 'r'&& str[2] == 'v')
                        return STD_OP_SRV;
```

Una volta definiti i dati, si aggiungere la gestione del filtro vero e proprio nel codice contenuto nel file *dumpstats.c*, modificando la sezione dedicata al comando "*show table*", facendo in modo di riconoscere il filtro del tipo `data.act_session srv <nomesrv>`: se l'operatore è "srv" salva il nome del server passato come parametro per potervi accedere successivamente:

```
if(s->data_ctx.table.data_op == STD_OP_SRV){
        strcpy(s->srvname,args[5]);
}
```

Nella funzione *stats_dump_table_to_buffer* è stata aggiunta nella prima parte una sezione di codice che scandisce i server presenti nel backend associato alla tabella selezionata, verifica se c'è una corrispondenza e manda un messaggio di errore se non è stato trovato nessun server. Per scandire tutti i server si deve utilizzare un ciclo in grado di scandire una lista di strutture di tipo server, sfruttando il campo "next" presente in ognuna di esse. Quando un server viene trovato, si applica un filtro sull'ID ad esso associato, così nella fase di dump il sistema automaticamente visualizzerà solo i dati richiesti.

```
/* if a filter for active session has been applies, search for the srv name and bind it with its id
 * otherwise return error
 */
if(s->data_ctx.table.data_op == STD_OP_SRV){
        struct proxy* px;
        px= s->data_ctx.table.target;
                //set comparation parameters to -1 in case there will be no matches
        s->data_ctx.table.value = -1;
                if(px != NULL){
                struct server* ser;
                s->data_ctx.table.data_op = STD_OP_EQ;
                /*added*/
                for(ser = px->srv;ser != NULL;ser=ser->next){
                        //check which server id to filter based on name
                        if(strcmp(s->srvname , ser->id) ==0){
                                s->data_ctx.table.value = ser->puid;
                                s->data_ctx.table.data_op = STD_OP_EQ;
                                printf("filter applied for server with name %s with id %d\n",
                                                ser->id,ser->puid);
                        }
                }
                /* no server has been found, send erros message*/
                if(s->data_ctx.table.value == -1){
                        chunk_printf(&msg, "No server Found with Name: %s\n",s->srvname);
                        if (buffer_feed_chunk(rep, &msg) >= 0)
                                        return 0;
                        s->data_state = DATA_ST_FIN;
                }
        }
}
```

Infine nella fase di filtraggio vero e proprio si è aggiunto il caso relativo al tipo di dato STD_T_STRING, che controlla se vi è un match tra il server ID precedentemente impostato e quello della entry da analizzare, se il valore è lo stesso, viene inviato un messaggio sul socket e l'operazione viene terminata.

```
/* compare values
 * if a match has been found, stop the search and print
 * the output message
 */
if(data == s->data_ctx.table.value){
        printf("session found! stopping the search\n");
        s->data_state = DATA_ST_FIN;
        //output message
        chunk_printf(&msg, "%s has still active sessions\n",s->srvname);
}
```

Questa nuova funzione è molto simile al comando show table, già presente in HAProxy, ma risulta essere più facile da utilizzare, dato che si deve semplicemente conoscere il nome di un server per poter accedere alle informazioni di cui si necessita, inoltre il tempo richiesto per eseguire questa operazione sarà sicuramente minore rispetto al dump dell'intera tabella, infatti la ricerca viene effettuata solo fino alla prima occorrenza utile, evitando così di dover accedere a tutti i dati della tabella.

Stick-store solo per server non-running

Un'altra modifica apportata al codice di HAProxy riguarda la gestione delle sticky table, per fare in modo che le sole sessioni associate a server in *maintenance mode* vengano inserite nella tabella. È importante sottolineare che senza questa modifica non sarebbe possibile avere la certezza che su un server in zombie mode non siano presenti sessioni attive. Le modifiche apportate al codice interessano le procedure di gestione delle sessioni e del parsing dei dati dei pacchetti ricevuti.

Nel file *session.c*, è stata modificata la sezione di codice in cui il load balancer decide se avviare la funzione per l'inserimento della entry nella sticky-table; è stata aggiunta una condizione che si verifica solamente se il server non è nello stato running.

```
struct server* srv= s->srv;
int state = srv->state;
if(!(state & SRV_RUNNING)){
```

Prima di applicare questa condizione è necessario accedere allo stato attuale del server corretto, tramite un puntatore della struttura session, che permette di accedere direttamente alla struttura del server in questione.

```
/* questa modifica attiva la sticky table solamente per i server che non
sono nella fase running
 * così si salveranno solamente le sessioni attive su un server in fase di
spegnimento*/
if(s->srv){
	struct server* srv= s->srv;
	int state = srv->state;
	if(!(state & SRV_RUNNING)){
		ts = stksess_new(rule->table.t, key);
		if (ts) {
			s->store[s->store_count].table = rule->table.t;
			s->store[s->store_count++].ts = ts;
		}
	}
}
```

Durante la fase di sperimentazione sono state effettuate ulteriori modifiche al codice sorgente di HAProxy, in primo luogo è stato modificato il codice che si occupa delle policy di inserimento delle entry nella sticky-table, facendo in modo che il sistema inserisca una entry per ogni richiesta ricevuta, anche se queste hanno la chiave in comune; tuttavia questa funzionalità non si è rivelata utile ai fini dell'implementazione del servizio di autoscaling.

Un'altra modifica è stata effettuata per poter aggiungere e togliere server da un backend inviando dei comandi tramite il socket di amministrazione, evitando quindi di dover riavviare il load balancer tutte le volte. Questa funzionalità può rivelarsi molto utile per uno sviluppo futuro del sistema, se si volesse rendere il load balancer in grado di mantenere tutti i dati in memoria. Per fare ciò si richiede di non

riavviare mai il processo, nemmeno in modalità soft-restart, il che rende fondamentali queste due funzioni.

9.4 Demone di Controllo – ElasticDaemon

Il *demone di controllo* sarà eseguito anch'esso sulla macchina che gestisce il cluster controller di Eucalyptus, questa scelta rende le interazioni con il load balancer e le istanze attive più semplici e sicure rispetto a un'installazione su macchine diverse. Il componente deve essere in grado di ricevere le informazioni fornite dall'utente, come tutte le richieste di inizializzazione e avvio di un cluster, o quelle di modifica dei parametri da utilizzare per gestire le diverse procedure, come ad esempio l'avvio di una nuova istanza o lo spegnimento di una macchina virtuale già attiva. Una volta elaborate tali istruzioni, il demone deve eseguire tutte le operazioni richieste, effettuando i controlli necessari affinché queste vadano a buon fine; si richiede quindi che il demone di controllo sia in grado di interagire con tutti i principali componenti del sistema: il load balancer, il monitor delle istanze e Eucalyptus.

Per interagire con HAProxy il demone modifica il file di configurazione utilizzato dal load balancer, aggiornandolo con le nuove informazioni, e rende effettive tali modifiche lanciando il comando di riavvio di HAProxy; viene utilizzato anche il socket unix messo a disposizione dal load balancer, per impostare alcuni parametri e per monitorare lo stato delle sessioni attive. Gli strumenti utilizzati per inviare richieste ad Eucalyptus sono costituiti dalla suite di comandi *euca2ools*, utilizzabili da shell, che possono interagire con il sistema IaaS avviando e spegnendo le istanze o monitorandone lo stato. Le informazioni relative all'utilizzo delle risorse, richieste e analizzate dal componente di monitoring, vengono ottenute tramite dei comandi inviati via *ssh* direttamente alle istanze attive, le quali devono essere dotate del software apposito per poter fornire le informazioni necessarie al monitor.

Il programma che implementa queste funzionalità è stato sviluppato in linguaggio *Java*, cercando di creare una struttura di oggetti e Thread che sia logicamente simile alla struttura del sistema di Amazon, in questo modo sarà possibile in futuro poter modificare o aggiungere le funzioni implementate seguendo quelle fornite da Amazon, senza dover modificare la struttura di base dell'intero programma.

9.4.1 Scaling Operation

Le procedure che deve svolgere ElasticDaemon sono riassumibili in quattro sequenze di operazioni ad alto livello, alcune delle quali devono essere svolte in contemporanea ad altre:

- *Scale-up*: Aggiunta di un'istanza al cluster
- *Scale-Down*: Spegnimento di un'istanza del cluster
- *Monitoring* delle istanze attive di un cluster
- *Avvio di un nuovo cluster* in base ai parametri forniti dall'utente

Ognuna di queste operazioni è stata implementata in un Thread separato, rendendo così le procedure delle scaling operation una indipendente dalle altre; ogni scaling operation viene avviata da un thread principale che conosce e controlla lo stato del cluster di istanze.

Scale-up Aggiunta di un'istanza

Questa Scaling Operation può essere invocata per tre motivi principali: in fase ai avvio del cluster, quando cioè nessuna istanza è attiva, di conseguenza per soddisfare il vincolo del minimo numero di istanze attive viene invocata la procedura di *Scale-up* fino a quando non sono state avviate un numero di macchine virtuali sufficiente; la procedura può essere attivata anche a seguito di una terminazione improvvisa di una delle istanze precedentemente attive, per sostituirla con una macchina virtuale funzionante, questa procedura deve essere preceduta da opportune operazioni per eliminare dal cluster l'istanza che ha terminato la sua esecuzione. I casi descritti rappresentano situazioni particolari in cui avviare la procedura di *scale-up*, che è invece destinata ad essere invocata quando si verifica che il cluster di istanze è sovraccarico, per aggiungere nuove risorse al fine di soddisfare le richieste in arrivo.

Le operazioni necessarie per completare correttamente la procedura di scale-up sono:

- Accedere alla *LaunchConfiguration* definita dall'utente in fase di avvio del cluster
- Verificare che siano disponibili risorse sufficienti su Eucalyptus per avviare una nuova istanza del tipo richiesto
- Avviare *una nuova macchina virtuale* utilizzando i parametri della launch configuration e memorizzare tutti i dati relativi alla nuova istanza avviata, come l'indirizzo IP privato e l'ID ad essa associati
- Verificare che il procedimento di avvio avvenga correttamente, interrogando Eucalyptus fino a quando l'istanza non sia in stato *"running"*. Questa fase può richiedere tempi abbastanza lunghi, infatti le operazioni di avvio effettuate all'interno di Eucalyptus possono richiedere attese dell'ordine dei minuti, a seconda della dimensione dell'istanza avviata e se questa sia già presente nella cache interna di Eucalyptus o meno
- Una volta terminata la fase di avvio dell'istanza si deve aggiungere l'entry ad essa relativa nel back end del load balancer, il sistema deve *aggiornare il file di configurazione* del load balancer aggiungendo i dati dell'istanza appena avviata
- Come ultima operazione si deve *riavviare il load balancer* per rendere effettive le modifiche apportate al file di configurazione

Terminate queste operazioni, le nuove richieste saranno automaticamente inviate anche alla nuova istanza, che diventerà attiva a tutti gli effetti per l'Autoscaling Cluster. Si è visto che è conveniente inserire una pausa di qualche decina di secondi prima di aggiungere i dati dell'istanza nella configurazione del load balancer, dato che Eucalyptus considera in stato *"running"* un'istanza da quando viene lanciato il sistema operativo; bisogna evitare che l'istanza venga aggiunta al cluster prima che abbia completato le proprie operazioni di boot e che abbia eseguito eventuali script di startup, visto che non potrebbe rispondere alle richieste inoltrate dal load balancer.

Scale-down Spegnimento di un'istanza

Questa scaling operation deve essere eseguita in due situazioni: può succedere che un'istanza del cluster termini inaspettatamente, o entri in uno stato che la rende inutilizzabile, in questo caso è necessario fare in modo che venga eliminata e che il traffico non venga più inoltrato ad essa; si libereranno inoltre le risorse che Eucalyptus aveva dedicato a questa istanza, per fare spazio ad una nuova macchina virtuale sostituiva. Il sistema dovrà poi invocare la *Scale-up* operation, in modo da ottenere un numero di istanze attive pari a quello che vi era prima della terminazione dell'istanza. Il caso più generale in cui viene invocata la *Scale-down* operation si presenta quando il monitor verifica che il cluster di istanze sta utilizzando un numero di risorse inferiore alla soglia minima definita dall'utente, si deve quindi procedere allo spegnimento di un'istanza, in modo da adattare le dimensioni del cluster al carico di lavoro. Questa operazione è molto importante per gli utenti, visto che consente di utilizzare solamente il numero di istanze strettamente necessario. È però cruciale definire correttamente le soglie da utilizzare per le decisioni, infatti se queste venissero definite con valori non adatti al tipo di applicazione utilizzato si possono verificare fenomeni di underprovisioning o addirittura si possono avviare dei "loop" che eseguono continuamente delle *Scale-down* operation seguite da *Scale-up* operation.

Le operazioni che si devono svolgere per completare la procedure da scale-down sono:

- *Scegliere quale istanza spegnere* tra quelle attive: si possono utilizzare due modalità, che risultano equivalenti ai fini del risultato finale. Si può scegliere un'istanza in maniera casuale tra quelle attive, quella attiva da più tempo o l'ultima che è stata avviata. Nell'implementazione si è scelto di spegnere l'istanza più recente
- Si deve poi impostare il load balancer in modo che il server dell'istanza scelta venga messo in modalità *"Zombie"*, così le nuove richieste verranno inviate alle altre istanze del cluster, mentre quelle relative a sessioni già associate al server verranno ancora inoltrate a quest'ultimo
- È necessario poi *attendere* un periodo di tempo pari al timeout di una sessione sull'application server, così si può avere la certezza che tutte le richieste che erano in fase di elaborazione nel momento in cui è stata attivata la *"Zombie"* mode del server sono terminate. Questa attesa si rende necessaria nel caso in cui si deve aspettare che il server invii una risposta a una richiesta che ha comportato delle elaborazioni molto lunghe, che possono durare al massimo un tempo uguale al session timeout impostato. Dopo questo periodo di pausa si ha la certezza che tutte le richieste che erano in fase di attesa quando il server è stato impostato in "zombie" mode sono terminate o correttamente, o con un messaggio di "session timeout" da parte del server
- A questo punto il sistema deve verificare che non ci siano ancora sessioni attive associate all'istanza da terminare. Si contatta ancora il load balancer utilizzando una delle funzionalità che sono state aggiunte, che consente di verificare se sono presenti sessioni attive su un determinato server. Se fossero ancora presenti sessioni attive, il processo dovrà controllare ad interrogare il load balancer fino a quando non siano tutte scadute per timeout. Dato che l'expire-time associato ad usa sessione viene reimpostato dopo ogni richiesta effettuata,

potrebbe accadere che questa operazione non termini mai, nel caso in cui una sessione venisse continuamente utilizzata; per evitare questo problema si potrebbe inserire un tempo massimo di attesa, superato il quale il sistema considererà tutte le sessioni scadute, indipendentemente dalle informazioni ricevute in merito dal load balancer. Tale soluzione può essere utile per evitare che utenti con comportamenti malevoli blocchino di proposito questo tipo di operazione

- Successivamente è possibile procedere con l'effettiva *rimozione dell'istanza dal cluster*, infatti nessuna richiesta sarà più inoltrata all'application server installato su di essa, di conseguenza spegnere questa macchina virtuale non comporterà nessun effetto visibile a livello utente. Per prima cosa si deve aggiornare il file di configurazione del load balancer, eliminando la entry relativa all'istanza da spegnere; una volta riavviato HAProxy con la nuova configurazione la macchina risulterà completamente isolata dalla rete esterna
- Si deve infine contattare Eucalyptus per *effettuare lo shutdown* vero e proprio della macchina; è sufficiente fornire l'ID dell'istanza che si vuole spegnere per avviare le operazioni di shutdown
- La scaling operation terminerà quando sarà verificato che lo stato dell'istanza sia passato da *"shutting-down"* a *"terminated"*

Questa serie di operazioni deve essere effettuata se l'utente richiede che si mantenga la persistenza delle sessioni a livello applicativo, mentre se viene richiesta la persistenza al solo livello di trasporto la procedura viene semplificata: non è necessario eseguire la fase di verifica per le sessioni ancora attive, basta solo aspettare che tutte le richieste attive terminino. Se invece la *scale-down* operation viene invocata quando un'istanza è già terminata accidentalmente o è comunque inutilizzabile, tutte le fasi iniziali possono essere eliminate, essendo l'istanza già compromessa. Si deve solo aggiornare il load balancer in modo che rimuova l'indirizzo della macchina virtuale in questione e terminarla, se necessario, invocando gli appositi comandi Eucalyptus.

Miglioramenti rispetto ad Amazon Autoscaling

Gestire la Scaling operation di Scale down seguendo la procedure precedentemente descritta permette ad ElasticDaemon di introdurre la funzionalità che si può definire *"session-aware scaling down"*, ossia il controllo completo dello stato delle sessioni quando il sistema deve spegnere una istanza. Il sistema sviluppato garantisce che tutte le sessioni attive non verranno mai terminate improvvisamente, a causa dello spegnimento del server che ne manteneva le informazioni. Gli utenti così non perderanno mai informazioni relative alle proprie sessioni, cosa invece possibile se si utilizza il servizio proposto da Amazon. Le procedure di *scale-down* infatti sono in grado di verificare se vi sono sessioni attive associate all'istanza che deve essere terminata e, se così fosse, ne posticipano lo spegnimento fino a quando tutte queste sono terminate; le operazioni eseguite da Amazon invece non verificano se vi sono sessioni attive, di conseguenza l'istanza viene terminata immediatamente, tuttavia in questo modo tutte le informazioni mantenute in memoria dall'istanza appena spenta vengono perse, obbligando gli utenti ad avviare delle nuove sessioni.

Un altro aspetto che rende il sistema proposto migliore rispetto alla soluzione proposta da Amazon riguarda i metodi con cui viene monitorato l'utilizzo delle risorse, infatti è possibile, sfruttando gli strumenti installati sulle virtual machines, ottenere informazioni riguardo a un maggior numero di parametri rispetto a quelli che si possono visualizzare tramite Amazon CloudWatch; è possibile quindi definire delle soglie di decisione più precise e elaborate in modo di poter avviare le scaling operation nei momenti più opportuni, in base alle caratteristiche dell'applicazione in esecuzione sulle istanze

Inizializzazione del load balancer e del cluster di VM

Prima di rendere operativo un cluster di istanze il sistema deve definire tutti gli oggetti necessari per l'esecuzione corretta di tutte le operazioni da effettuare. Le operazioni da eseguire sono del tutto simili a quelle che si utilizzano nella definizione di un Autoscaling Cluster con i servizi Amazon. Molte delle informazioni che vengono inserite durante la fase di inizializzazione possono essere modificate successivamente, senza compromettere l'esecuzione delle istanze o interrompere le scaling operation attive. Le operazioni da eseguire sono le seguenti:

- *Definire i parametri del load balancer* da utilizzare, in particolare l'indirizzo IP e la porta su cui verranno ricevute le richieste in ingresso, il nome del cookie da elaborare, l'intervallo di tempo dopo cui una sessione viene considerata scaduta dall'application server e il metodo di bilanciamento da utilizzare. Una volta definite queste operazioni il sistema inizializza i parametri base di configurazione del load balancer generando il nuovo file di configurazione e prepara il file binario di HAProxy per l'esecuzione.
- *Definire la launch configuration* da utilizzare quando si devono avviare nuove istanze. I parametri da definire sono il tipo di istanza e l'immagine da utilizzare, oltre informazioni aggiuntive come gli eventuali user-data e la chiave da utilizzare per accedere all'istanza tramite ssh.
- *Definire il monitor e i trigger*. Va definita per prima cosa la metrica da utilizzare, tra utilizzo di CPU, quantità di dati scritti o letti su disco e quantità di traffico ricevuto o inviato; si inseriscono poi i dati corrispondenti alle soglie che devono essere superate per attivare le scaling operation per aumentare e diminuire il numero di istanze attive nell'Autoscaling Cluster.
- Si può infine *avviare il cluster*, fornendo gli ultimi parametri fondamentali, che corrispondono al numero minimo e massimo di istanze attive. Una volta completata questa operazione il sistema potrà avviare le prime istanze e rendere il cluster operativo.

Una volta che il cluster è stato attivato questa serie di operazioni non verrà più eseguita, sarà possibile solamente modificare alcuni dei parametri in base a specifiche richieste dell'utente. Un esempio tipico riguarda la modifica di parametri relativi alla *launch configuration* utilizzata, che consente di avviare istanze con funzionalità e potenza diverse senza dover eseguire nuovamente tutte le operazioni di inizializzazione.

Monitoring delle risorse utilizzate dalle istanze

Queste operazioni vengono eseguite da un processo che ripete le stesse procedure periodicamente, in base ai parametri definiti dall'utente. Ad ogni esecuzione il sistema deve contattare tutte le istanze e ottenere i dati necessari per aggiornare lo stato di utilizzo delle risorse dell' Autoscaling Cluster. Le operazioni che vanno eseguite sono:

- *Accedere alla lista di tutte le istanze attive*: è necessario che questa operazione venga effettuata in maniera sicura, non deve infatti accadere che si acceda ad una lista quando un'operazione concorrente, come ad esempio una Scale-down Operation, sta modificando gli oggetti in essa contenuta. Una volta ottenuta la lista di istanze da monitorare si può passare alla fase di recupero delle informazioni
- Per ogni istanza attiva si deve *interrogare l'applicativo* `sar` , in grado di fornire le informazioni sull'utilizzo di risorse, installato su ognuna delle macchine. L'output ottenuto va analizzato per estrarre i parametri da memorizzare per fornire i dati statistici ad ElasticDaemon. `sar` viene invocato tramite ssh, fornendo oltre all'indirizzo IP dell'istanza anche la chiave da utilizzare per ottenere l'accesso, in modo da non dover inserire una password. L'esecuzione di questo comando richiede un tempo di attesa che corrisponde circa all'intervallo di tempo scelto per la misurazione delle risorse con il tool sar. Non è comunque fondamentale che i risultati vengano ottenuti immediatamente, dato che il periodo con cui le istanze devono essere interrogate è dell'ordine dei minuti.
- Una volta ottenuti tutti i dati relativi alle istanze in esecuzione, il monitor deve *aggregare i dati*, secondo quanto è stato definito nella fase di inizializzazione del cluster. Il valore risultante può essere il valore medio, il minimo, il massimo o la somma dei valori singoli ottenuti tramite sar.

Le operazioni eseguite dal monitor è conveniente che vengano eseguite in un processo separato rispetto agli altri, consentendo quindi al sistema di avere aggiornamenti costanti sullo stato del sistema, anche quando sono attive scaling operation, che interromperebbero il servizio di monitoring per intervalli di tempo troppo lunghi, rendendo più difficile poter valutare lo stato globale del cluster di istanze.

10. IMPLEMENTAZIONE DI ELASTICDAEMON

Il servizio *ElasticDaemon* è stato implementato in linguaggio *Java*, mantenendo lo stile di Eucalyptus, visto che i componenti del frontend sono stati sviluppati utilizzando lo stesso linguaggio di programmazione. Java si presta anche facilmente ad essere integrato con altri servizi, come il supporto a ricevere e inviare richieste a servizi web, oltre ad essere in grado di gestire in maniera efficiente l'esecuzione di Thread concorrenti.

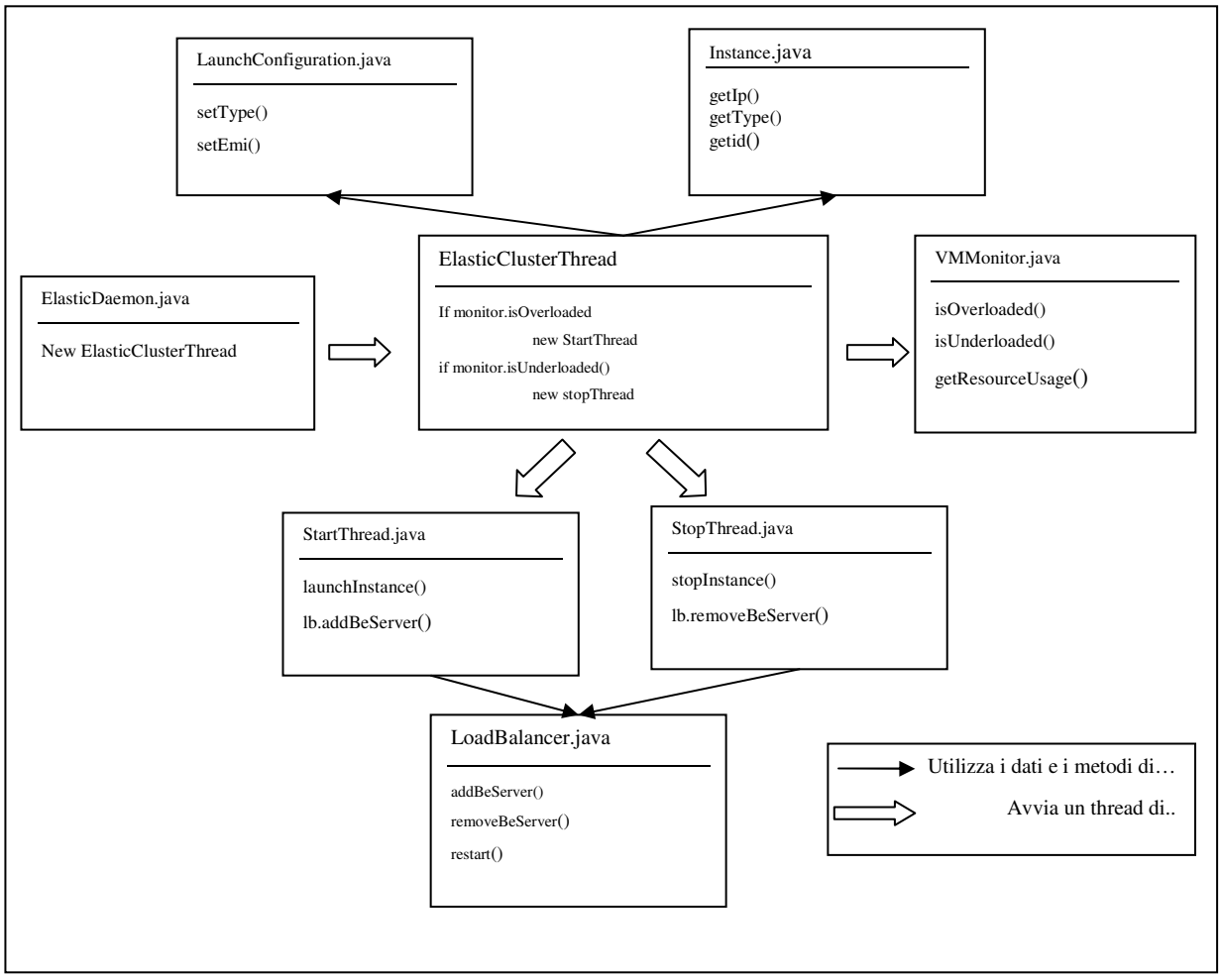


Figura 34 Schema delle classi Java sviluppate

L'architettura di *ElasticDaemon* è costituita da una serie di classi Java, di cui alcune implementano l'interfaccia Runnable e verranno eseguite come thread, mentre altre sono utilizzate per la memorizzazione e l'accesso dei dati necessari all'esecuzione dell'Autoscaling Cluster. Il programma è in grado di avviare e gestire più autoscaling cluster indipendenti tra loro, in modo da poter soddisfare le richieste di tutti gli utenti di Eucalyptus. La struttura base che è stata utilizzata per sviluppare l'intero programma è rappresentata in Figura 34.

10.1 Workflow del programma

10.1.1 ElasticDaemon

La classe eseguibile principale è *ElasticDaemon*, che si occupa di inizializzare tutti gli oggetti necessari per poter rendere attivo il cluster di istanze. Quando tutte le operazioni di inizializzazione sono state eseguite, ossia tutti gli oggetti richiesti sono stati creati, la classe crea un oggetto di tipo *ElasticClusterThread,* fornendo come parametri gli oggetti precedentemente definiti. Il ruolo della classe principale a questo punto è terminato per il cluster in questione, potrà essere utilizzata per gestire eventuali modifiche da effettuare ai parametri impostati nella prima fase.

Quando gli oggetti vengono creati, vengono effettuate alcune operazioni iniziali al fine di garantire il funzionamento corretto del sistema: in particolare vengono preparati i file necessari per l'esecuzione di HAProxy, creando un file di configurazione relativo al nuovo cluster e creando una copia del file eseguibile, che sarà poi utilizzato quando il cluster di istanze diventerà operativo. Un'ultima operazione che il sistema deve eseguire è quella di aggiungere l'indirizzo IP pubblico del load balancer all'interfaccia di rete pubblica della macchina, in questo modo si possono ricevere i pacchetti che hanno come indirizzo di destinazione quell'IP, i quali a loro volta saranno poi gestiti dal load balancer.

10.1.2 ElasticClusterThread

Il Thread appena avviato, della classe *ElasticClusterThread,* contiene tutte le funzioni richieste per gestire in maniera autonoma il comportamento dell'Autoscaling Cluster: il thread è costituito da un loop infinito, che continuamente verifica lo stato delle istanze attive e, in base al carico di lavoro riscontrato, ha il compito di attivare le scaling operation opportune. Nella fase di inizializzazione vengono indicati tutti gli oggetti che concorreranno nella gestione dell'insieme di istanze, in particolare vengono memorizzati un oggetto di tipo LoadBalancer, uno di tipo monitor e una lista di oggetti *Instance*. Prima di entrare nel loop infinito il thread avvia il processo di monitoring, costituito anch'esso da un thread, di classe VMMonitor, che verrà descritto più dettagliatamente in seguito. Una volta avviato il thread di monitoring il cluster inizia a controllare se sono verificate le condizioni per avviare o, in alternativa spegnere, una istanza del cluster; se nessuna di queste condizioni risulta vera, il thread attende un intervallo di tempo di qualche secondo e poi riesegue i controlli sulle condizioni; la pausa che viene effettuata tra un ciclo e l'altro è stata inserita per non sovraccaricare inutilmente la macchina che esegue questo programma, infatti non è necessario che il thread di controllo risponda in realtime ad un

cambiamento dello stato del cluster, dato che le stesse scaling operation richiedono tempi di attesa dell'ordine di qualche minuto. Conviene quindi mantenere il sistema un po' meno reattivo, ma che utilizza molte meno risorse, in particolare la CPU.

Oltre a monitorare se le condizioni per avviare una scaling operation siano verificate, ad esempio se il sistema è sovraccarico, si deve avviare una Scale-up operation, il thread di controllo del cluster deve controllare anche se vi siano altre operazioni attive in quell'istante, se questo ulteriore controllo non venisse effettuato, può succedere che più scaling operation uguali vengano attivate in sequenza, facendo aumentare, o diminuire, troppo velocemente il numero delle istanze attive.

Basti pensare al caso in cui si verifichi un picco di richieste, ad ogni iterazione la condizione per attivare una nuova scale-up operation sarebbe vera, si avvierebbero così in numero imprecisato di nuove istanze, senza considerare che si stanno già svolgendo le operazioni di startup di altre macchine virtuali. Questo fenomeno non solo è logicamente scorretto, ma sarebbe anche poco gradito agli utenti, che dovrebbero pagare per un numero troppo elevato di istanze attive, quando la quantità richiesta per soddisfare le richieste in arrivo sarebbe molto inferiore.

10.1.3 VMMonitor

Questa classe si occupa di monitorare continuamente lo stato di utilizzo di risorse delle istanze, gli oggetti di questa classe sono eseguiti come Thread, infatti ad ogni Cluster di istanze viene associato un thread di classe VMMonitor. Le operazioni svolte da questo processo sono abbastanza semplici: il thread è costituito da un ciclo infinito, all'interno del quale viene scandita la lista contenente le informazioni sulle istanze attive, ad ognuna di esse viene inviata la richiesta per i dati sull'utilizzo delle risorse. Quando tutte le informazioni sono state raccolte, queste vengono elaborate secondo quanto definito nella fase di inizializzazione, calcolando i valori massimi, medi, ecc. Infine vengono aggiornati i valori generali, che rappresentano lo stato di utilizzo di risorse da parte del cluster; tali variabili sono quelle che poi vengono confrontate con le soglie definite dall'utente, per decidere se il cluster di istanze è sovraccarico di richieste o meno.

Il thread di monitor contiene, oltre alle istruzioni per l'esecuzione del thread principale, una serie di metodi per l'accesso ai valori raccolti, in particolare per confrontare i valori con le soglie di massimo e minimo utilizzo, come i metodi *isOverloaded e isUnderloaded*.

10.1.4 LoadBalancer

Questa classe si occupa di interagire e di gestire il funzionamento del load balancer associato all'Autoscaling Cluster; ad ogni oggetto ElasticClusterThread è associato un oggetto *LoadBalancer*, che contiene tutti i parametri e fornisce tutti i servizi per gestire l'esecuzione dell'istanza di HAProxy ad essa dedicata.

Quando viene inizializzato un oggetto di tipo *LoadBalancer*, viene per prima cosa effettuata una copia dell'eseguibile di HAProxy, che si occuperà del bilanciamento del traffico tra le istanze. Oltre

all'eseguibile viene creato un file di testo contenente le informazioni generali che deve conoscere il load balancer, come la politica di gestione delle sessioni, l'algoritmo di bilanciamento e l'indirizzo IP pubblico e la porta su cui dovrà ascoltare il frontend di HAProxy. L'istanza di HAProxy non viene avviata fino a quando non è presente almeno un'istanza attiva in grado di rispondere alle richieste in arrivo sul frontend, visto che per poter completare le procedure di startup lo stesso HAProxy richiede che venga definito almeno un server di backend. Gli oggetti di tipo LoadBalancer non sono eseguibili come thread, ma forniscono solo metodi di supporto che vengono invocati durante le scaling operation avviate dalla classe che si occupa della gestione e dell'esecuzione del cluster di istanze.

Le principali funzioni che fornisce la classe LoadBalancer sono:

- *Avvio e riavvio* in modalità soft dell'istanza di HAProxy
- *Modifica del file di configurazione* in base ai nuovi parametri e alle nuove istanze attive
- *Invio di comandi al load balancer*, come il comando per impostare un server di backend in modalità "Zombie"
- Richiesta dei *dati statistici sulle sessioni*, per verificare se vi sono ancora sessioni attive su un determinato server di backend

10.1.5 Instance

Questo tipo di oggetto rappresenta un'istanza Eucalyptus, insieme a tutti i dati ad essa correlati; il sistema mantiene in memoria lo stato attuale di ogni istanza, insieme a tutti quei dati che vengono assegnati a runtime ad ogni istanza da Eucalyptus come l'ID univoco assegnato alla virtual machine e l'indirizzo IP ad essa associato. Gli oggetti *Instance* contengono poi anche tutti gli altri parametri caratteristici di un'istanza, come il tipo di macchina e il codice dell'immagine utilizzata per l'esecuzione; ogni oggetto fornisce i classici metodi *get* e *set* per i parametri appena descritti; al momento della creazione dell'oggetto non tutti i valori dei parametri sono conosciuti, infatti l'indirizzo IP e l'ID di una particolare istanza sono noti solo dopo che sono state completate le procedure di startup della macchina virtuale sul nodo di Eucalyptus.

10.1.6 Launch Configuration

Ogni oggetto ElasticClusterThread deve mantenere in memoria un puntatore ad un oggetto *LaunchConfiguration*, che contiene i parametri che il sistema deve utilizzare quando deve richiedere l'avvio di una nuova istanza ad Eucalyptus; l'oggetto *LaunchConfiguration* mantiene in memoria anche la access key e la secret key necessarie per potersi autenticare durante le interazioni con il servizio di IaaS.

I dati mantenuti in memoria corrispondono a tutti i parametri di avvio di una nuova macchina virtuale, ossia l'immagine da utilizzare, il tipo di macchina, la chiave di accesso da inserire per i collegamenti ssh e eventuali user-data che si vogliono passare all'istanza.

10.1.7 StartThread e StopThread

Queste due classi vengono utilizzate per eseguire le *scaling operation* per aggiungere e togliere un'istanza dal cluster; non espongono particolari metodi pubblici; entrambe implementano l'interfaccia *Runnable*, che consente di avviare thread a partire da queste classi. Il metodo principale è *run()*, invocato durante l'esecuzione del thread che contiene tutto il codice necessario per eseguire correttamente tutte le procedure di Scale-up, nel caso di StartThread, e di Scale-Down, nel caso di StopThread; sono presenti altri metodi privati definiti per rendere l'implementazione più efficiente. Ognuno di questi Thread riceve come parametro un puntatore all'oggetto ElasticClusterThread che lo ha invocato; utilizzando i metodi forniti da tale oggetto, i Thread possono avere accesso a tutte le informazioni necessarie per completare correttamente le operazioni che hanno il compito di svolgere.

10.2 Aspetti Pratici

Il sistema sviluppato deve interagire con componenti eterogenei, come il frontend di Eucalyptus, il load balancer HAProxy e le istanze in esecuzione; è stato necessario utilizzare tecniche diverse per eseguire tutte le diverse operazioni che sono state descritte nei paragrafi precedenti.

10.2.1 Interazione con il frontend di Eucalyptus

Il frontend di Eucalyptus costituisce l'unico punto di accesso per poter visualizzare lo stato delle istanze attive, monitorare la quantità di risorse disponibili e inviare le richieste di avvio e di spegnimento di istanze. Tutte queste operazioni vengono utilizzate in quasi ogni procedura implementata per realizzare il servizio di autoscaling.

I metodi con cui si può interagire con Eucalyptus sono due, equivalenti tra loro: si può inviare una *richiesta ad un servizio web*, seguendo le definizioni dello standard utilizzato (che è lo stesso di Amazon), oppure utilizzare la suite di *comandi eseguibili da shell euca2ools*, il cui ruolo non è altro che quello di costruire le richieste in maniera appropriata e inviarle al sistema. La soluzione utilizzata per lo sviluppo di ElasticDaemon è quella di invocare i comandi euca2ools tramite la classe Process di java, che consente di eseguire comandi linux passati come stringa al metodo `exec(String cmd)`. Utilizzando i comandi forniti da euca2ools non solo non è necessario dover costruire la richiesta seguendo lo standard del servizio web, ma si può anche verificare facilmente se la richiesta è andata a buon fine analizzando l'output prodotto sulla linea di comando. La classe Process infatti fornisce anche il metodo *getInputStream()* che consente di ricevere e successivamente analizzare i dati dell'output del comando eseguito.

I comandi euca2ools vengono invocati dai due thread StartThread e StopThread, sono infatti questi che hanno il compito di interagire con Eucalyptus, verificando la disponibilità di risorse, avviando le istanze e monitorandone lo stato. Grazie ai dati che si possono ricavare utilizzando questi comandi il sistema di autoscaling riesce ad ottenere tutte le informazioni di cui necessita per garantire la regolare gestione di ogni Autoscaling Cluster definito dagli utenti. Per eseguire questi comandi sono necessari alcuni dati relativi alle credenziali dell'utente, infatti Eucalyptus verifica l'autenticità di ogni richiesta richiedendo

che vengano forniti sempre le chiavi di accesso e le chiavi segrete dell'utente. A questi dati sono contenuti nella launch configuration associata all'autoscaling cluster, consentendo così al sistema di poter invocare i comandi euca2ools fornendo le credenziali di accesso corrette.

10.2.2 Interazione con HAProxy

Il sistema deve interagire con il load balancer utilizzando modalità differenti: deve essere in grado di *modificare i file di testo* di configurazione e salvarli sul disco, *avviare il file eseguibile* di HAProxy in modalità hot-reconfiguration, per riavviare il load balancer in modo da rendere attive le eventuali modifiche appostate al file di configurazione e deve anche *poter inviare delle richieste sul socket unix* di amministrazione associato ad ogni istanza di HAProxy.

Per accedere, modificare e aggiornare i file di testo che contengono i file di configurazione vengono utilizzate le classi Java del package `java.io`, che permettono di leggere e scrivere facilmente dati su file. È stato sfruttato anche il metodo `deleteOnExit()` per rendere i file creati solamente temporanei; tutti i file su cui è stato invocato il metodo citato verranno infatti cancellati dal disco una volta che l'esecuzione del programma termina. Questa tecnica implementativa risulta utile per evitare di utilizzare una quantità di spazio eccessiva su disco, visto che tutti i file creati hanno nomi diversi, ognuno associato al nome del cluster a cui fa riferimento; utilizzando questa tipologia di implementazione si evita di dover creare delle funzioni che si occupano di cancellare i file obsoleti.

Per avviare il server e inviare i comandi di *hot-reconfiguration* è stato utilizzato anche in questo caso il metodo *exec()* della classe Process; i comandi eseguiti richiedono anche un accesso di root, è necessario quindi precedere il comando con il prefisso sudo, oppure modificare il file di configurazione di HAProxy per abilitare la funzione di chroot. Il fatto di dover avviare questi comandi da superuser non è da considerarsi una limitazione implementativa troppo stringente, infatti queste operazioni devono essere eseguite sulla macchina del Cluster Controller di Eucalyptus, di cui l'amministratore ha il pieno controllo; per mantenere anche un alto livello di sicurezza si può definire un nuovo utente che abbia i privilegi di root per eseguire solamente i comandi richiesti da ElasticDaemon. Per effettuare un *soft-restart* di HAProxy bisogna non solo disporre del file di configurazione aggiornato, ma è necessario conoscere anche il numero del pid associato all' istanza attiva del load balancer.

Per inviare comandi e ricevere le risposte utilizzando il socket unix si può utilizzare il tool *socat*, che permette di interagire con i socket unix inviando i comandi utilizzando un particolare comando da shell del tipo: `echo "<command>"| sudo socat -s unix-connect:<socketfile> stdio`

Per eseguire comandi di questo tipo non è sufficiente fornire come parametro la stringa che si digiterebbe su una shell, infatti il metodo exec della classe Process non interpreta i caratteri particolari come se fossero stati digitati da una shell. Nel caso del comando sopra descritto, i caratteri che simboleggiano “pipe” e i doppi apici utilizzati per racchiudere il comando non vengono interpretati correttamente e quindi il comando non viene eseguito ccome rischiesto. Per ovviare a questo problema è stato sviluppato uno script bash, denominato `launcher.sh`, che riceve come parametri tutti i token

dell'intero comando che si vuole eseguire, sarà poi lo script a riconoscere i caratteri particolari e ad eseguire il comando come richiesto dall'utente.

Questa soluzione introduce un passaggio aggiuntivo durante l'esecuzione dei comandi, tuttavia per Java non è presente una libreria che si riesca ad interfacciare con il socket unix di HAProxy, si è visto quindi che l'utilizzo dello script bash per l'esecuzione effettiva dei comandi è il metodo più diretto per superare le limitazioni del metodo *exec()*.

10.2.3 Interazione con le Istanze in Esecuzione

Per poter monitorare l'utilizzo di risorse delle istanze in esecuzione, si è scelto di interrogare un programma installato direttamente sul sistema operativo delle macchine virtuali stesse. Questo tipo di operazioni vengono eseguite dagli oggetti di tipo *VMMonitor*, che utilizzano il metodo exec per avviare una connessione ssh con l'istanza; per stabilire una connessione senza dover inserire manualmente una password, si fornisce come parametro al comando ssh una chiave che la macchina è in grado di riconoscere, autenticando la richiesta e attivando la connessione.

Oltre al certificato viene indicato anche il comando da eseguire sull'istanza, che invoca il tool `sar` di sysstat con i parametri scelti in accordo con le metriche che si devono misurare. L'output risultante viene poi analizzato per estrarre i valori richiesti. Visto che viene utilizzato un accesso ssh alla macchina, nell'immagine utilizzata si devono definire le chiavi di accesso per rendere possibile il collegamento alla macchina da parte del sistema di autoscaling. Per mantenere livelli di sicurezza elevati, si può definire un accesso con privilegi ridotti, in grado solamente di eseguire i tool di monitoring di sysstat.

10.3 Analisi del Codice sorgente

In questa sezione verranno descritte le sezioni più significative delle classi Java sviluppate, in particolare evidenziando come sono state implementate le procedure precedentemente descritte.

10.3.1 ElasticDaemon.java

Questa classe ha il ruolo di definire tutti gli oggetti necessari all'avvio del cluster, sono presenti quindi delle istruzioni di inizializzazione, seguite dall'avvio vero e proprio del Thread che gestisce un cluster.

```
/* perform initialization activities */
String clusterName = "Cluster1";
/*initialize a load balancer*/
LoadBalancer lb = new LoadBalancer(clusterName,"192.168.20.254",5001,
                                   “JSESSIONID",60,LoadBalancer.ROUNDROBIN);
/*initialize launch configuration*/
LaunchConfiguration launchConfiguration = new LaunchConfiguration("Cluster1",
                              "emi-03F71116",LaunchConfiguration.M1_SMALL,"test");
/*initialize the monitor*/
VMMonitor monitor = new VMMonitor("Monitor",70,20,
        VMMonitor.CPUUTILIZATION,VMMonitor.AVERAGE,5);
/* initialize and start an autoscaling cluster*/
Thread cluster = new Thread(new ElasticClusterThread(clusterName,lb,launchConfiguration,1,10,monitor));
cluster.start();
```

L'istruzione più importante è l'ultima, che lancia il thread di controllo del cluster "*Cluster1*" appena definito; i parametri utilizzati sono tutti quelli descritti nei paragrafi precedenti, in questo caso viene avviato un load balancer che ascolterà sulla porta 5001 dell'indirizzo ip pubblico 192.168.20.254, l'algoritmo di bilanciamento è di tipo roundrobin e il cookie da analizzare è JSESSIONID, mentre le sessioni sull'application server hanno durata di 60 secondi. Verranno avviate da una a un massimo di 10 istanze di tipo m1.small, la chiave di accesso da utilizzare per le connessioni ssh è test.key.

Questa configurazione è stata utilizzata per verificare il corretto funzionamento del sistema, tutti i parametri sono realistici, tranne il valore utilizzato per la durata delle sessioni, che solitamente è di circa 15 minuti, tuttavia questo non influisce sul comportamento del sistema, visto che tale parametro determina solamente i tempi che alcune operazioni devono attendere prima di essere eseguite.

10.3.2 ElasticClusterThread.java

Ogni oggetto di questa classe rappresenta un cluster di istanze: è l'elemento che conosce l'intero stato dell'insieme di macchine virtuali e ha il compito di avviare le scaling operation quando si verificano le condizioni appropriate. Gli oggetti di questo tipo vengono definiti fornendo al costruttore tutti i puntatori agli oggetti che rappresentano il load balancer, la launch configuration da utilizzare e il monitor che verrà utilizzato per richiedere alle istanze la quantità di risorse che stanno utilizzando; vengono inoltre indicati altri parametri quali il nome del cluster e il numero minimo e massimo di istanze che si possono avviare. Il costruttore non esegue operazioni particolari, ma si limita a memorizzare i dati che sono stati forniti come parametri.

```
public ElasticClusterThread(String clusterName, LoadBalancer lb,LaunchConfiguration
                launchConfiguration,int minSize,int maxSize,VMMonitor monitor) throws Exception{
        /* initializes the thread object and the monitor*/
        this.clusterName=clusterName;
        ...
        System.out.println("Created Thread " + clusterThread.getName());
}
```

La classe implementa il metodo run, che viene invocato quando il thread associato ad essa viene fatto partire; in questo metodo sono implementate tutte le istruzioni necessarie per la gestione del cluster, in grado di scegliere se e quando avviare le scaling operation opportune. La struttura del metodo si divide in due parti: nella prima viene definito e lanciato il thread che si occuperà di monitorare le istanze del cluster, quando questo processo è stato attivato, continuerà a interrogare le istanze attive, quando presenti, e a memorizzare i dati raccolti, fornendo poi i metodi per accedere ai dati raccolti. Nella parte iniziale del metodo run viene anche definito un Thread "control" che verrà utilizzato per le operazioni successive.

```
/*start the monitor thread*/
monitor.setInstances(instances);
Thread mon = new Thread(monitor);
mon.start();
Thread control = null;
```

La seconda parte del metodo è costituita da un loop infinito, nel quale verranno continuamente effettuati i controlli per verificare se il cluster richiede un aumento o una diminuzione del numero delle istanze. Ogni volta che una condizione è verificata viene creato un nuovo thread, che si occuperà di eseguire le procedure della scaling operation da svolgere; si deve però evitare che si attivino più operazioni contemporaneamente, per questo si sono definiti i metodi *setDoingOperations()* e *isDoingInstanceOperations()*, che impostano e verificano il valore di una variabile di controllo.

Ogni volta che si deve attivare una Scaling Operation si deve verificare che la variabiledi controllo sia *false*, confermando che non siano attive altre procedure, si procede quindi a invocare il metodo *setDoingOperations(true)* per informare il sistema che è in corso una scaling operation; infine si attiva effettivamente il thread associato alla procedura che si vuole svolgere.

Per poter verificare che il thread appena attivato sia andato a buon fine ed abbia terminato la sua esecuzione, viene creato un thread control che ha il compito di verificare quando la scaling operation ha concluso le sue operazioni; questo thread dopo aver atteso la terminazione dell'esecuzione, utilizzando il metodo *join()* della classe Thread, attende un intervallo di tempo predefinito, che consente all'intero cluster di stabilizzarsi, prima di terminare la sua procedura.

```
while(true){
try{
if(control!= null && !control.isAlive())
        setDoingInstanceOperations(false);

/* if the cluster is overloaded or the active instances are less than minSize, launch a new instance
if((getNumInstances() < minSize || (monitor.isOverloaded() &&
                        getNumInstances()<maxSize))&& !isDoingInstanceOperations()){
                        setDoingInstanceOperations(true);
        ….
        /* launch the instance and sets the corresponding ID and IP
         * after that it informs the lb to route traffic on the new instance */
        Thread launch = new Thread(new LaunchThread(newInstance, this));
        launch.start();
        ...
}
if (monitor.isUnderloaded() && getNumInstances() > minSize &&
                        isDoingInstanceOperations()){
                        setDoingInstanceOperations(true);
        …..
        /*puts the lb entry of the instance in  "zombie" mode, and waits for all the sessions
         * active to expire, after that is removes the entry from the lb and stops the instance
         */
        Thread stop = new Thread(new StopThread(instance,this));
        stop.start();
        ….
}
Thread.sleep(1000);
```

All'interno del ciclo while vengono effettuati tre diversi controlli, eseguiti uno di seguito all'altro:

- il primo controllo verifica se il *Thread control è terminato*, se questo è vero significa che non ci sono scaling operation attive, ossia è possibile invocare il metodo *setDoingOperations(false)* dando l'autorizzazione ad eseguire una nuova procedura, qualora fosse necessario.
- Il secondo controllo verifica se le condizioni di utilizzo del cluster richiedono che venga attivata una nuova istanza. I casi per cui questa situazione si può verificare sono sostanzialmente due: non sono attive sufficienti istanze nel cluster, per cui bisogna attivarne una di nuova per soddisfare il vincolo del numero minimo di istanze attive, oppure le risorse utilizzate dalle istanze in esecuzione hanno superato la soglia massima e il numero di istanze attive non è maggiore del numero massimo consentito. Ad ognuna di queste condizioni si aggiunge il controllo sull'eventuale presenza di altre scaling operation attive, che impedisce di avviarne di nuove; questa situazione si può facilmente presentare quando una scaling operation dello stesso tipo è in esecuzione ma deve ancora terminare. È molto probabile infatti che quando il sistema è sovraccarico mantenga questo stato fino a quando una nuova istanza è attiva, può succedere quindi che il controllo sull'utilizzo di risorse del sistema richieda di attivare una nuova istanza, ma l'operazione non venga eseguita visto che una analoga è in esecuzione. I metodi utilizzati per decidere se attivare una nuova istanza sono associati all'oggetto di tipo VMMonitor, come *isOverloaded()* che dice se l'utilizzo di risorse è superiore alla soglia massima, mentre il metodo *getNumInstances()* viene utilizzato per controllare se il numero di istanze attive è consistente con i vincoli imposti.
- Il terzo controllo è speculare al secondo, viene infatti invocato il metodo di VMMonitor *isUnderloaded()* per verificare se le risorse utilizzate sono inferiori al limite minimo scelto dall'utente. Se questa condizione è verificata e il numero di istanze è maggiore del minimo numero richiesto, viene avviata la *scale-down* operation che si occupa di eliminare un'istanza dal cluster. Ovviamente anche in questo caso è possibile avviare l'operazione solamente se non ne sono attive delle altre.

Alla fine del ciclo è presente una *pausa*, che consente al sistema di non essere troppo sovraccaricato da un eccessivo numero di controlli. Il loop continuerà ad essere eseguito fino a quando l'utente non chiederà che il cluster venga eliminato dal sistema.

10.3.3 Thread Control

Ogni volta che viene avviata una scaling operation, viene creato anche un *thread di controllo* per verificare la corretta terminazione dell'operazione avviata. Le istruzioni che effettuano queste operazioni instanziano un nuovo oggetto di tipo *ControlThread* e gli forniscono come parametro il thread della scaling operation. L'oggetto appena creato viene assegnato alla variabile control, di tipo Thread, che viene utilizzata nella prima verifica eseguita nel loop principale del processo dell'oggetto ElasticClusterThread.

```
launch.start();
/* wait for the launch thread to finish
 */
control = new Thread(new ControlThread(launch));
control.start();
```

L'implementazione del metodo *run* di ControlThread è molto semplice, infatti l'unica operazione che deve eseguire è quella di verificare che il thread che ha ricevuto come parametro termini la sua esecuzione, per poi terminare a sua volta. Il metodo utilizzato per attendere il termine dell'esecuzione del thread da controllare è *join()* della classe Thread.

```
toWait.join();
Thread.sleep(WAIT);
System.out.println("Operation thread finished");
```

10.3.4 Mutua esclusione

È stato necessario proteggere gli accessi all'oggetto che contiene la lista di istanze attive sul cluster (di tipo ArrayList<Instances>), infatti tale elemento viene acceduto da diversi processi concorrenti. Innanzitutto vi accede il thread di monitor, che ad ogni periodo deve verificare quante istanze sono attive e contattarle per ottenere le informazioni sull'utilizzo di risorse. L'oggetto viene modificato anche dai thread che eseguono le scaling operation, i quali devono aggiungere o togliere oggetti di tipo *Instance* dalla lista, e infine lo stesso thread principale del cluster vi accede per sapere il numero di istanze attive.

Per questo ogni blocco di codice che deve accedere o modificare l'oggetto instances è stato racchiuso in una sezione che richiede l'accesso in mutua esclusione alla lista di istanze. È stato utilizzato il costrutto di java *synchronized(oggetto){codice}*. A titolo di esempio si riporta il codice che implementa il metodo addInstance(Instance i) della classe ElasticClusterThread:

```
public void addInstance(Instance instance){
        synchronized(instances){
                instances.add(instance);
        }
}
```

Questa sintassi consente di evitare di ottenere stati inconsistenti del sistema, oltre a non generare eccezioni quali ConcurrentModificationException, lanciata dalla classe ArrayList se si verifica che vengono effettuate modifiche concorrenti allo stesso oggetto. È stata protetta da accessi in mutua esclusione anche la variabile *doingOp*, di tipo Boolean; questa soluzione per il momento non è strettamente necessaria, visto che tale variabile viene acceduta solamente dalla classe ElasticClusterThread, tuttavia i metodi ad essa associati potrebbero essere richiamati in altre situazioni, di conseguenza si è preferito aggiungere tale protezione per facilitare eventuali sviluppi futuri.

10.3.5 LaunchThread.java

Questa classe implementa tutte le procedure per l'esecuzione della scaling operation che ha il compito di aggiungere una nuova istanza al cluster. Quando un nuovo oggetto di tipo *LaunchThread* viene creato, sarà possibile poi utilizzarlo per avviare un nuovo thread, che avrà il compito di avviare una nuova istanza ed aggiungerla al cluster. Il costruttore riceve come parametri l'oggetto di tipo Istanza, che rappresenta la macchina virtuale che dovrà essere avviata, e l'oggetto di tipo ElasticClusterThread, dal quale potrà accedere a tutte le informazioni sul cluster.

```
public LaunchThread(Instance instance, ElasticClusterThread cluster) {
        /* store private variables */
        launchThread = new Thread(instance.getEmi());
        this.instance=instance;
        this.cluster = cluster;
        System.out.println("Created Launch Thread for emi: " + launchThread.getName());
}
```

Il metodo principale di questa classe è *run()*, che viene invocato quando viene avviato il thread ad esso associato. In questo metodo sono realizzate tutte le procedure necessarie per garantire la corretta inizializzazione della nuova macchina virtuale.

Per prima cosa il metodo verifica se sul Eucalyptus sono disponibili risorse sufficienti per avviare una nuova macchina, invocando il metodo *isAvailable()*, il quale ritorna un valore booleano che indica se sia possibile avviare una nuova istanza. A questo punto viene invocato il metodo che si occupa del lancio dell'istanza su Eucalyptus, bisogna fornire come parametri il tipo di macchina da avviare e il nome dell'immagine da utilizzare. Quando il metodo *launchInstance* termina, è garantito che l'istanza è stata correttamente avviata e si trova nello stato "running", si può procedere quindi alla modifica del file di configurazione del load balancer e a riavviarlo per rendere effettive le modifiche.

```
System.out.println("Launching thread started...");
/*if there are available resources on the cloud, start a new instance */
if(isAvailable()){
        launchInstance(instance.getEmi(),instance.getType());
}else{
        System.out.println("Not enough space for the new Instance of type: " + instance.getType());
        return;
}
/*get the load balancer of the cluster */
LoadBalancer lb = cluster.getLoadBalancer();
/* add the new instance to the load balance cfg file*/
lb.addBeServer(instance.getId(), instance.getIp(), 8080 , cluster.getClusterName());

/* restart the load balancer to route the traffic on the new instance */
lb.restart();

/* add the Instance to the cluster */
cluster.addInstance(instance);
```

Per eseguire le operazioni appena descritte si utilizzano i metodi forniti dall'oggetto di tipo *LoadBalancer*; prima viene invocato il metodo *addBeServer()*, che richiede come parametri l'indirizzo IP e la porta su cui ascolterà l'application server installato, insieme all'ID univoco dell'istanza e al nome del cluster a cui appartiene; una volta terminata l'esecuzione di questo metodo il file di configurazione è aggiornato e contiene tutte le informazioni sul nuovo server installato sull'istanza appena avviata. Infine si invoca il metodo restart della classe LoadBalancer, che riavvia HAProxy aggiornandone la configurazione in base al nuovo file appena creato.

isAvailable()

La realizzazione del metodo *isAvailable()* consiste in due fasi: invocazione del comando *euca-describe-availability-zones* e l'analisi dell'output ottenuto. Per utilizzare il comando euca-describe-availability-zones viene invocato il metodo runCommand di una classe si supporto, che riceve come parametro una stringa contenente il comando da eseguire, e ritorna l'output ottenuto. Tale metodo di supporto utilizza la funzione exec della classe Process di Java.

```
StringBuilder cmdOut = CommandsHandler.runCommand("euca-describe-availability-zones "+
        "verbose " +
        " -a " + cluster.getLaunchConfiguration().getEc2AccessKey() +
        " -s " + cluster.getLaunchConfiguration().getEc2SecretKey() +
        " -U " + cluster.getLaunchConfiguration().getEc2Url());
String regexp = "AVAILABILITYZONE\\s+\\S+\\s+"+instance.getType()
                        + "\\s+(\\d+) / \\d+\\s+";
Pattern pattern = Pattern.compile(regexp);
Matcher matcher =pattern.matcher(cmdOut);
int count=0;
while(matcher.find()){
        count++;
        if(Integer.parseInt(matcher.group(1))>0)
                return true;
}
if(count ==0){
        System.out.println("Error in parsing euca-describe-availability-zones output");
        System.exit(1);
}
return false;
```

Dopo aver lanciato il comando, l'output ottenuto deve essere analizzato per verificare se sono disponibili le risorse necessarie, per fare questo si utilizza una regular expression che estrae il valore relativo al numero di istanze avviabili per un determinato tipo di macchina. Bisogna anche verificare questo valore su tutti gli eventuali cluster di nodi disponibili su Eucaluptus, per questo si è introdotto un ciclo che scandisce tutti i possibili match ottenuti dall'espressione regolare. Il metodo *isAvailable()* ritorna true se il numero estratto è maggiore di zero, false altrimenti. Un tipico output che si deve elaborare è il seguente:

```
bazzu@ubuntu-bazzu:~$ euca-describe-availability-zones verbose
AVAILABILITYZONE  cluster1    192.168.10.1
AVAILABILITYZONE  |- vm types free / max   CPU   ram  disk
AVAILABILITYZONE  |- m1.small 0007 / 0008   1    192     2
AVAILABILITYZONE  |- c1.medium      0007/ 0008   1    256     5
AVAILABILITYZONE  |- m1.large       0002 / 0004   2    512     10
AVAILABILITYZONE  |- m1.xlarge      0001 / 0003   2   1024     20
AVAILABILITYZONE  |- c1.xlarge      0001 / 0001   4   2048     30
```

In questo esempio il valore che viene estratto dall'espressione regolare, nel caso in cui il tipo di macchina richiesto sia m1.small è 0007, che indica la disponibilità di risorse per altre 7 istanze di quel tipo.

LaunchInstance

Questo metodo ha il compito di avviare una nuova istanza su Eucalyptus, successivamente ne deve estrarre alcuni dei parametri univoci che il sistema le associa, come l'indirizzo IP privato e l'ID, infine deve verificare che lo stato dell'istanza passi da “pending” a “running”. *LaunchInstance* conclude la sua esecuzione quando verifica che l'avvio della nuova istanza è andato a buon fine. Per avviare una nuova istanza si esegue il comando *euca-run-instances*, fornendo tutti i parametri necessari per l'autenticazione e per la definizione dell'istanza che si deve avviare.

```
/* command euca-run-instances */
StringBuilder cmdOut = CommandsHandler.runCommand("euca-run-instances "+
                    " -a " + cluster.getLaunchConfiguration().getEc2AccessKey() +
                    " -s " + cluster.getLaunchConfiguration().getEc2SecretKey() +
                    " -U " + cluster.getLaunchConfiguration().getEc2Url()+
                    " -k " + cluster.getLaunchConfiguration().getKeyName()+
                    " --addressing private" +
                    " " +emi);
```

Il comando euca-run-instances, quando completa correttamente la sua esecuzione, fornisce in output l'ID univoco associato alla nuova istanza attivata; si deve quindi analizzare il risultato che si ottiene dal comando lanciato e estrarre da esso questo identificativo, che sarà fondamentale per impostare tutti gli altri parametri del sistema. Per estrarre l'ID dall'output viene invocato il metodo *getIdFromLaunch()* che riceve come parametro l'oggetto contenente l'output ottenuto, successivamente viene analizzato con una regular expression che estrae i dati richiesti, infine il valore dell'ID viene ritornato come stringa.

```
private String getIdFromLaunch(StringBuilder input){
        String regexp = "INSTANCE\\s+(i-\\w*)\\s+" + instance.getEmi();
        Pattern pattern = Pattern.compile(regexp);
        Matcher matcher =pattern.matcher(input);
        if(matcher.find())
                return matcher.group(1);
        else{
                System.out.println("Error during launch of instance");
                System.exit(1);
        }
}
```

L'ultima operazione che deve effettuare il metodo *launchConfiguration()* è quella di verificare che Eucalyptus completi le procedure di startup dell'istanza senza errori, si utilizza un ciclo che continua ad invocare il metodo *isRunning()*, che dice se l'istanza è nello stato "running" o meno.

Se l'istanza non è ancora entrata nello stato running, viene invocato il metodo *checkStateAndGetIp()*, che ha il compito di verificare lo stato dell'istanza, in più tale metodo deve estrarre dall'output ottenuto l'indirizzo IP che è stato assegnato all'istanza. Questa informazione deve essere ottenuta tramite il comando euca-describe-instances, infatti Eucalyptus assegna gli indirizzi ip alle macchine in un istante successivo al processo di inizializzazione, non è quindi possibile ottenere l'indirizzo di rete di una macchina se non quando è già stata avviata la sua procedura di startup. All'interno del ciclo è stata inserita una pausa, dell'ordine di qualche secondo, per evitare di sovraccaricare troppo il sistema e di inviare un numero eccessivo di richieste ad Eucalyptus.

```
while(!isRunning()){
        checkStateAndGetIp();
        Thread.sleep(CHECK_INTERVAL);
}
```

Il metodo invocato nel ciclo appena descritto esegue due operazioni principali: esegue da linea di comando *euca-describe-instances* <*id*> utilizzando l'identificatore che è stato ottenuto precedentemente. Una volta ricevuto l'output da Eucalyptus ne elabora il contenuto utilizzando una *regular expression*, che verifica se lo stato della macchina è running e ne estrae l'indirizzo di rete ad essa assegnato.

```
private void checkStateAndGetIp() throws IOException{
        StringBuilder cmdOut = CommandsHandler.runCommand("euca-describe-instances "+instance.getId()+
                                                " -a " + cluster.getLaunchConfiguration().getEc2AccessKey() +
                                        " -s " + cluster.getLaunchConfiguration().getEc2SecretKey() +
                                        " -U " + cluster.getLaunchConfiguration().getEc2Url());
                getRunningStateAndIp(cmdOut);
        }

private void getRunningStateAndIp(StringBuilder input){
        String regexp = "INSTANCE\\s+"+ instance.getId() + "\\s+" + instance.getEmi() +
        "\\s+((\\S*|\\.*)*)\\s+((\\S*|\\.*)*)\\s+(\\w*)";
        Pattern pattern = Pattern.compile(regexp);
        Matcher matcher =pattern.matcher(input);
        if(matcher.find()){
                if(matcher.group(5).equals("running")){
                        isRunning = true;
                        instance.setIp(matcher.group(3));
                        return;
                }else if(matcher.group(4).equals("terminated")){
                        System.out.println("Instance "+instance.getId() + " failed to startup");
                        System.exit(1);
                }
        }
}
```

Il metodo *launchInstance()* introduce, prima di concludere la sua esecuzione, una pausa di qualche decina di secondi, in questo modo si consente al sistema operativo installato sulla macchina virtuale di completare il processo di boot. A questo punto l'avvio della nuova istanza è stato completato, si dispone di una nuova macchina virtuale in stato running, di cui il sistema conosce tutti i dati necessari per renderla effettivamente attiva nel cluster, ossia aggiungendola alle entry del load balancer, che inizierà ad inoltrarle le richieste degli utenti.

Operazioni finali di aggiornamento

Le ultime operazioni da svolgere per il *LaunchThread* sono quelle di aggiornamento del file di configurazione e del riavvio del server; l'implementazione di queste funzioni verrà descritta nel dettaglio quando verrà analizzata la classe LoadBalancer.java.

Come ultima istruzione il LaunchThread deve comunicare al Cluster la presenza di una nuova istanza attiva, che contribuirà a rispondere alle richieste inoltrate dal load balancer; per fare ciò viene invocato il metodo *addInstance(Instance i)* della classe ElasticClusterThread, bisogna sottolineare che l'aggiornamento della lista delle istanze avviene in mutua esclusione, al fine di garantire che non vi siano errori durante l'esecuzione degli altri thread concorrenti, come quello di monitoring

10.3.6 LoadBalancer.java

Ogni oggetto di tipo *LoadBalancer* rappresenta un'istanza attiva del load balancer e fornisce le funzioni necessarie per interagire con HAProxy. Un oggetto di questa classe è sempre associato ad uno di tipo ElasticClusterThread, che rappresenta il cluster di istanze su cui il load balancer bilancerà il traffico che riceverà in ingresso sul frontend.

Per inizializzare un oggetto di tipo LoadBalancer si devono fornire diversi parametri al costruttore della classe, quali indirizzo IP e porta di ascolto del frontend, il nome del cluster di istanze associato, oltre ai parametri per la gestione della session stickiness a livello 7. Il costruttore della classe, oltre a memorizzare i parametri ricevuti, crea il file di configurazione e una copia dell'eseguibile di HAProxy.

```
public LoadBalancer(String clusterName, String fe_ip, int fe_port,
        String cookieName,int sessionTimeout,int balanceAlgorithm) throws IOException{
        ...
        createCfgFile(clusterName,cookieName,balanceAlgorithm);
        /*prepare the executable file */
        copyHAProxy("HAProxy_"+clusterName);
}
```

Il metodo *createCfgFile()* costruisce un file di configurazione a partire da alcuni pattern preimpostati, personalizzandoli con i parametri forniti al costruttore della classe, mentre *copyHAProxy()* effettua una copia del file eseguibile con il nome: HAProxy_<nomecluster>.

I metodi più importanti di questa classe, che vengono invocati durante le scaling operation sono quelli che permettono di aggiungere o rimuovere un server dal file di configurazione, di riavviare il load

balancer in modalità soft-restart, di interrogare il load balancer per sapere se su un determinato server sono ancora attive delle sessioni e infine per impostare un server in modalità “zombie”.

AddBeServer e removeBeServer

Questi due metodi effettuano operazioni opposte sul file di testo di configurazione di HAProxy, il primo legge il file di configurazione da disco, scandisce il contenuto fino a trovare la posizione in cui aggiungere l'entry del server e inserisce una nuova riga in quel punto; successivamente il file di configurazione viene salvato su disco, sovrascrivendo la versione precedente. Il secondo metodo effettua la stessa sequenza di operazioni, ma al posto di inserire una nuova entry, elimina quella relativa al server indicato come parametro del metodo. Verrà di seguito descritta solo l'implementazione del metodo *addBeServer()*, che risulta molto simile a quella di *removeBeServer()*, essendo le procedure svolte molto simili.

Per prima cosa il metodo invoca una funzione di supporto per ottenere i dati dal file di configurazione, restituisce un oggetto di tipo StringBuilder, contenente tutto il testo letto dal file. Si deve poi trovare la posizione corretta in cui inserire la nuova riga che descrive il nuovo server di backend da aggiungere, anche in questo caso viene utilizzata un regular expression che trova come corrispondenza le entry del backend.

```
public void addBeServer(String srvId, String srvIp, int srvPort, String clusterName)throws IOException{
        StringBuilder cfg = getCfgFile();
        /* search for the correct backend section*/
        String regexp = "\\t#"+clusterName + " backend servers:\\n(\\tserver \\S* "
                        + REGEX_IP + ":\\d* cookie \\S*\\n)*";
        Pattern pattern = Pattern.compile(regexp);
        Matcher backendSrv =pattern.matcher(cfg);
        int numMatches=0;
        int end=0;
        while(backendSrv.find()){
                numMatches++;
                end=backendSrv.end();
        }
        /* if there is no match, or more than one, show error message */
        if(numMatches == 0 ||numMatches >1){
                System.out.println("Error while finding backend servers for cluster "+clusterName);
                System.exit(1);
        }
}
```

Una volta trovato un match nel file di configurazione si deve estrarre l'indice in cui si deve inserire la nuova riga, si utilizza il metodo *end()* della classe matcher, che ritorna l’indice dell'ultimo carattere che corrisponde alla regular expression. Il formato che si ricerca all'interno del file di configurazione è il seguente:

```
#Cluster1 backend servers:
      server i-233243 192.168.20.20:8080 cookie i-233243
      server i-456234 192.168.20.21:8080 cookie i-456234
```

Per facilitare il compito della regular expression è stato inserita una riga "sentinella" (#Cluster1 backend servers:) che permette di individuare più facilmente le entry dei server di backend all'interno del file di configurazione. A questo punto è sufficiente aggiungere la nuova entry con i parametri dell'istanza e salvare il nuovo file su disco.

```
/*insert the new server in the correct backend*/
cfg.insert(end, "\tserver "+srvId+ " " + srvIp +":" + srvPort + " cookie " + srvId+"\n");

/* write the cfg file to disk */
writeToCfg(cfg);
```

hasActiveSessions

Questo metodo viene invocato durante le procedure di *scale-down*, viene utilizzato per interrogare il load balancer e per sapere se sono presenti sessioni attive su un determinato server. Per interagire con il load balancer è necessario inviare una stringa del tipo: *show table <nomebackend> data.act_session srv <nomeserver>* sul socket unix su cui ascolta l'istanza del load balancer. I parametri che devono essere passati a questo metodo sono semplicemente il nome del server di cui si vogliono ricevere le informazioni e il nome dell'autoscaling cluster a cui è associato il load balancer. Per inviare tale comando al load balancer si utilizza il già citato metodo exec(), utilizzando però lo script di bash launcher.sh che è in grado di eseguire i comandi sfruttando le funzioni della shell, come il carattere speciale "pipe".

```
public boolean hasActiveSessions(String srvName, String clusterName)throws IOException{
        /* launch the command on the socket to verify if there are still active sessions for the instance
         *  example:
         *  echo "show table tomcat data.act_session srv web1" |sudo socat -s unix-connect:/tmp/HAProxy stdio
         */
        StringBuilder input = CommandsHandler.runCommand("./launcher.sh " +
                "echo show table "+clusterName + "_backend data.act_session srv " + srvName +" "
                                        + PIPE +" " +SOCAT_CMD + lbSocket + " stdio");

        /*parse output*/
        String regexp = "has still active sessions";
        Pattern pattern = Pattern.compile(regexp);
        Matcher matcher =pattern.matcher(input);
        if(matcher.find())
                return true;
        else
                return false;
}
```

L'output viene poi analizzato e si verifica la presenza della stringa "has still active sessions", se questa è presente il metodo ritorna true, altrimenti false.

SetZombie

Il metodo *setZombie()* viene invocato durante la scale-down operation, per comunicare al load balancer di non inoltrare ulteriori richieste a tal server. I parametri richiesti sono anche in questo caso il nome del

server e il nome del cluster, necessari per identificare correttamente la coppia backend/server da impostare in modalità zombie. Il metodo deve inviare un comando al socket unix del tipo: *disable server <namobackend>/<nomesserver>*; si utilizza il metodo exec, sfruttando lo script bash launcher.sh come già descritto nel metodo *hasActiveSessions()*.

```
public void setZombie(String srvName, String clusterName) throws IOException{

        StringBuilder cmdOut = CommandsHandler.runCommand("./launcher.sh " +"echo disable server "
        +clusterName + "_backend/"+srvName +" "+PIPE+" " + SOCAT_CMD + lbSocket + " stdio");

        String regexp = "\\w+"; //any output is an error message
        Pattern pattern = Pattern.compile(regexp);
        Matcher matcher =pattern.matcher(cmdOut);
        if(matcher.find()){
                System.out.println("Error while disabling server: " + srvName +
                                                        " in cluster: " + clusterName);
                System.exit(1);
        }
}
```

Il comando disable server non mostra nessun output se l'operazione va a buon fine, di conseguenza il metodo si limita a verificare che nell'output non sia presente alcun carattere, se accade tale evento viene inviato un messaggio di errore, altrimenti il metodo termina correttamente.

10.3.7 StopThread.java

Questa classe implementa tutte le procedure necessarie per eseguire la *scale-down* operation, come nel caso di StartThread viene implementata l'interfaccia Runnable, in modo da poter inizializzare un Thread ogni volta che si vuole eliminare un'istanza dal cluster. Un thread di tipo *StopThread* viene attivato dalla classe ElasticClusterThread, quando si verifica che le istanze del cluster stanno utilizzando una quantità di risorse troppo bassa e si richiede quindi che una di esse termini la propria esecuzione. Per inizializzare un oggetto *StopThread* si devono passare come parametri l'oggetto di tipo Istanza che rappresenta la virtual machine che si vuole spegnere e l'oggetto di tipo ElasticClusterThread che contiene tutte le informazioni dell'Autoscaling Cluster a cui appartiene l'istanza. Il costruttore si limita a memorizzare su variabili private i parametri ricevuti; la funzione principale della classe è, come nel caso di StartThread, il metodo run().

Run

Questo metodo ha il compito di implementare tutta quella serie di operazioni che sono necessarie ad eseguire la scale down operation. Per prima cosa si rimuove l'istanza dalla lista di quelle attive sul cluster, in questo modo il thread di monitoring non richiederà più le informazioni sull'utilizzo di risorse a tale virtual machine; viene poi invocato il metodo *setZombie()* dell'oggetto load balancer, che si occuperà di interagire con HAProxy per attivare la modalità “zombie” del server installato sull'istanza. In seguito a questa chiamata il thread attende che tutte le comunicazioni attive siano terminate, effettuando una pausa pari all'expire time di una sessione sull'application server.

```
/* remove the instance from the cluster arraylist */
cluster.removeInstance(instance);

/*get the loadbalancer of the cluster*/
LoadBalancer lb = cluster.getLoadBalancer();
/*put the instance in zombie mode*/
lb.setZombie(instance.getId(),cluster.getClusterName());

/*wait for the timeout of all the sessions*/
Thread.sleep(lb.getSessionTimeout()*1000);
```

Dopo aver atteso che tutte le connessioni correnti siano terminate, si deve controllare se sul server sono ancora attive delle sessioni a livello applicativo, questa operazione avviene invocando il metodo *hasActiveSessions()* della classe LoadBalancer, fornendo come parametri l'ID dell'istanza che corrisponde al nome del server per il load balancer e il nome del cluster. Se il risultato di tale chiamata è positivo, StopThread continua a richiamare questo metodo fino a quando tutte le sessioni hanno raggiunto l'expire-time. Come nei casi precedenti, per evitare un eccessivo numero di richieste e per non sovraccaricare il sistema viene effettuata una pausa di qualche secondo tra un controllo e l'altro.

Quando tutte le sessioni sono terminate, si può procedere a rimuovere l'entry del server dal file di configurazione del load balancer, invocando il metodo *removeBeServer()*, indicando l'ID dell'istanza e il nome del cluster, e riavviare HAProxy per rendere attive le modifiche.

```
/* check if there are still active sessions */
while(lb.hasActiveSessions(instance.getId(),cluster.getClusterName())){
        Thread.sleep(CHECK_INTERVAL);
}
/*remove the server from the loadbalancer cfg file*/
lb.removeBeServer(instance.getId(),cluster.getClusterName());
/* restart the load balancer to apply the changes */
lb.restart();

/*shut down the instance*/
terminateInstance(instance);
```

Le operazioni preliminari per lo spegnimento dell'istanza sono ora completate, infatti l'application server installato risulta totalmente isolato dalla rete esterna, visto che non vi sono più sessioni attive su di esso e il load balancer non ha più la entry ad esso associata tra i server di backend. È possibile procedere con la fase di spegnimento dell'istanza, invocando il metodo interno alla classe *terminateInstance*, che richiede come parametro l'oggetto di tipo istanza che corrisponde alla macchina virtuale da spegnere. Questa funzione interagisce con Eucalyptus per inviare il segnale di spegnimento dell'istanza, e poi verifica che l'operazione sia andata a buon fine.

terminateInstance

Il metodo *terminateInstance()* esegue due operazioni: la prima esegue il comando *euca-terminate-instance <id>* che consente di avviare la fase di spegnimento della macchina virtuale di cui è stato indicato l'ID.

```
/*run command to terminate the instance */
CommandsHandler.runCommand("euca-terminate-instances "+instance.getId()+
                        " -a " + cluster.getLaunchConfiguration().getEc2AccessKey() +
                        " -s " + cluster.getLaunchConfiguration().getEc2SecretKey() +
                        " -U " + cluster.getLaunchConfiguration().getEc2Url());
/* wait for shutdown */
while(!checkTerminateState()){
        Thread.sleep(CHECK_INTERVAL);
}
```

Come per tutti gli altri comandi inviati da shell, anche in questo caso viene utilizzata la funzione exec di Java. Una volta inviato il segnale di spegnimento il metodo deve verificare che le risorse allocate per l'istanza siano state effettivamente liberate; per fare questo si deve aspettare che lo stato dell'istanza passi da "*shutting-down*" a "*terminated*", solo in questo momento Eucalyptus considererà concluso il ciclo di vita della macchina virtuale, e libererà le risorse allocate.

Per verificare lo stato dell'istanza viene invocato il metodo checkTerminateState, il quale esegue il comando *euca-describe-instances <id>* e analizza l'output ottenuto, e ritorna true se lo stato è "terminated" false altrimenti. La verifica dell'output avviene utilizzando una regular expression che estrae lo stato rilevato dell'istanza, che poi viene confrontato con la stringa target.

```
/* runs euca-describe-instances */
StringBuilder cmdOut = CommandsHandler.runCommand("euca-describe-instances "+instance.getId()+

                        " -a " + cluster.getLaunchConfiguration().getEc2AccessKey() +
                        " -s " + cluster.getLaunchConfiguration().getEc2SecretKey() +
                        " -U " + cluster.getLaunchConfiguration().getEc2Url());
/* parse the output and checks if the instance is terminated */
String terminated_regex = "INSTANCE\\s+"+ instance.getId() + "\\s+" + instance.getEmi() +
"\\s+\\S+\\s+\\S+\\s+terminated";
Pattern pattern = Pattern.compile(terminated_regex);
Matcher matcher =pattern.matcher(cmdOut);
if(matcher.find())
        return true;
else
        return false;
```

10.3.8 VMMonitor.java

La classe VMMonitor si occupa di interagire da un lato con le istanze attive per ricevere le informazioni dal tool *sar*, mentre dall'altro deve comunicare al cluster se le risorse utilizzate sono superiori alle soglie definite.

La classe implementa l'interfaccia Runnable, per poter creare un thread che monitora in continuazione le istanze attive, indipendentemente dalle altre scaling operation che potrebbero essere in esecuzione sul

cluster. Per facilitare l'utilizzo della classe sono state definite alcune variabili pubbliche non modificabili, che rappresentano i tipi di metriche che si possono misurare e quale tipo si statistica utilizzare per valutare la quantità di risorse utilizzata:

```
public static final int AVERAGE =0;
public static final int MAX     =1;
public static final int MIN     =2;
public static final int SUM     =3;

public static final int CPUUTILIZATION              =0;
public static final int DISKUTILIZATION             =1;
public static final int NETWORKINUTILIZATION  =2;
public static final int NETWORKOUTUTILIZATION       =3;
```

Il Costruttore della classe richiede una lunga serie di parametri,necessari a definire come il monitor deve interagire con le istanze e quali metodi statistici deve valutare, inoltre vanno definite le soglie, minima e massima, che determineranno se il cluster si deve considerare sovraccarico o meno. Le operazioni eseguite dal costruttore memorizzano tutta la lista di parametri nelle variabili provate, per essere poi accedute per le elaborazioni successive.

Il metodo *run* è costituito da un loop infinito, che richiede alle istanze la quantità di risorse che stanno utilizzando e ne salva i risultati; una volta completate queste operazioni attende per un periodo di tempo personalizzabile dall'utente, per poi ricominciare la propria attività.

Per monitorare le istanze si deve accedere alla lista di macchine virtuali attive sul cluster, che deve essere acceduta in mutua esclusione; tutto il primo blocco di codice quindi è protetto dall'istruzione *synchronized(instances)*. Per ogni istanza viene invocato il metodo *getResourceUsage()*, che ritorna il valore di utilizzo della risorsa richiesta. Mentre viene eseguito il ciclo che monitora le istanze una ad una, vengono calcolati alcuni semplici dati statistici, utili per le elaborazioni successive, quali il valore massimo, minimo e il totale delle misurazioni effettuate.

```
long value=0;
int total=0; //total of the values
long min= Integer.MAX_VALUE; //minimum value
long max = Integer.MIN_VALUE; //maximum value
synchronized(instances){
        for(Instance i : instances){ /* for each instance get resource usage */
                try{
                        value = getResourceUsage(i);
                        /* update the sum and the min-max values */
                        total+=value;
                        if(value >=max)
                                max=value;
                        if(value < min)
                                min=value;
                }catch(IOException e){
                        System.out.println("Error monitoring instance " +i.getId());
                        System.exit(1);
                }
                System.out.println("Instance: " +i.getIp() + " usage: " + value);
        }
```

Terminata la fase di reperimento dei dati, vengono calcolati i valori che devono essere memorizzati in base al parametro che definisce quale tipo di dato statistico utilizzare. Infine il valore risultante viene memorizzato per renderlo accessibile tramite i metodi isOverloaded e isUnderloaded. Infine il thread deve attendere un intervallo di tempo pari al periodo di monitoring impostato dall'utente, per poi rieseguire le operazioni descritte.

```
if(instances.size()>0){
        switch(statistics){
        case AVERAGE:
                actualResourceUsage = total/instances.size();
                break;
        case MAX:
                actualResourceUsage = max;
                break;
        case MIN:
                actualResourceUsage = min;
                break;
        case SUM:
                actualResourceUsage = total;
        }
}
...
public boolean isOverloaded(){
        return actualResourceUsage > THRESHOLD_HIGH;
}
public boolean isUnderloaded(){
        return actualResourceUsage < THRESHOLD_LOW;
}
```

La variabile *actualResourceUsage* è dichiarata *volatile*, in quanto viene acceduta e scritta da più thread concorrenti, utilizzando tale tipo di dichiarazione ogni accesso a questa variabile avverrà come se si trovasse in un blocco synchronized, proteggendo quindi l'integrità dei valori ad essa associati.

11. CONCLUSIONI E SVILUPPI FUTURI

In questa tesi sono state presentate le caratteristiche e le funzionalità dei servizi di C*loud Computing Infrastructure-as-a-Service*, evidenziando quali vantaggi si possono ottenere grazie all'utilizzo di questo tipo di risorse e quali problematiche si devono affrontare. Nella prima parte si è visto come vi sono delle applicazioni che si adattano particolarmente ad essere sviluppate ed eseguite utilizzando servizi *cloud* di tipo IaaS, come, ad esempio, tutte quelle che richiedono un buon livello di interattività con l'utente e che hanno carichi di lavoro difficilmente prevedibili. Successivamente sono stati analizzati tutti i fattori che si devono valutare per decidere se sia conveniente utilizzare risorse IaaS, considerando non solo gli aspetti economici, ma anche le problematiche legate alla migrazione delle applicazioni e di come questo cambiamento possa essere percepito dagli utilizzatori delle stesse.

Nella seconda parte della tesi è stato analizzato un prodotto open source, *Eucalyptus*, in grado di fornire un servizio simile a quelli offerti da Amazon EC2, descrivendo nel dettaglio l'architettura e il funzionamento dell'applicazione. Successivamente sono state evidenziate tutte le funzionalità che tale software è in grado di fornire e quelle che invece non sono ancora state implementate: tra queste non è presente il servizio di *Autoscaling*, che consente di definire un *cluster* di macchine virtuali, in grado di aumentare e diminuire le proprie dimensioni in base al carico di lavoro da svolgere.

È stato progettato e realizzato un servizio di Autoscaling per *Eucalyptus*, che non solo comprende tutte le funzionalità della soluzione proposta da Amazon, ma è anche in grado di gestire in maniera affidabile la persistenza delle sessioni a livello applicativo. Ai componenti base di *Eucalyptus* sono stati aggiunti tre moduli aggiuntivi: un *load balancer*, per gestire il bilanciamento del carico di lavoro; un *monitor*, per controllare lo stato di utilizzo delle risorse; un *demone di controllo*, per eseguire le *scaling operation*. Il sistema è in grado di gestire tutte le operazioni di aumento e diminuzione del numero di istanze del cluster e di informare il load balancer perché possa inoltrare le richieste degli utenti alle macchine virtuali disponibili. Una delle capacità che contraddistingue il software sviluppato è quella di poter monitorare le sessioni attive a livello applicativo, agendo in modo da evitare la perdita dei dati di sessione degli utenti durante le *scaling operation*. Utilizzando il servizio proposto, uno sviluppatore può avviare la propria applicazione su varie istanze "indipendenti" tra loro, evitando così di dover gestire le

eventuali interazioni tra di esse, semplificando le operazioni di configurazione del sistema, senza dover rinunciare a fornire ai propri clienti un servizio scalabile e affidabile.

I servizi di *Cloud Computing* nei prossimi anni svolgeranno sicuramente un ruolo importante nel settore dell'*Information Technology*, sia per quanto riguarda le offerte *Software-as-a-Service*, grazie alla sempre più avanzata tecnologia dei client e alla disponibilità di collegamenti veloci, sia per i servizi *Infrastructure-as-a-Service*, che risultano molto interessanti per le aziende e gli sviluppatori di applicazioni. È emerso infatti che le risorse *cloud* di tipo infrastrutturale possono essere vantaggiose dal punto di vista economico per molte attività, in particolare per tutte quelle che operano in settori strettamente collegati al Web. I principali vantaggi che un'azienda può trarre dall'utilizzo di servizi *cloud* sono innanzitutto una riduzione dei costi operativi e un annullamento degli investimenti iniziali, ma anche la garanzia di dover pagare esattamente in base alla quantità di risorse effettivamente utilizzate. Quest'ultimo aspetto è di fondamentale importanza, soprattutto perché il numero di risorse richieste da un'applicazione è strettamente collegato al numero di clienti che utilizzano un determinato servizio e, di conseguenza, ai ricavi che si potranno ottenere. Oltre ai vantaggi economici, vi sono anche delle agevolazioni dal punto di vista del mantenimento dei sistemi, infatti tutta la gestione "fisica", con le problematiche del caso, viene delegata a terzi, come la collocazione, l'alimentazione e la manutenzione delle macchine.

Una delle problematiche che emerge quando si sceglie di utilizzare i servizi cloud riguarda le garanzie che vengono rilasciate sulla disponibilità di servizio: nonostante vengano assicurate percentuali di *uptime* molto elevate, i *cloud provider* non sono ancora in grado di garantire una disponibilità totale, per questo motivo i servizi IaaS non possono essere utilizzati per applicazioni che non ammettono periodi di *downtime*, come quelle che operano in ambito governativo o finanziario. È da sottolineare però che nella maggior parte dei casi le garanzie offerte dai *cloud provider* soddisfano pienamente le richieste degli sviluppatori: di solito viene assicurato un *uptime* del 99.9% del servizio durante il corso dell'anno, riuscendo così a rendere i servizi IaaS un'ottima alternativa a soluzioni di hosting tradizionali. Un altro aspetto che devono considerare gli utenti è rappresentato dai livelli di sicurezza garantiti dal provider, che devono essere verificati e confrontati con le necessità dell'applicazione; oltre a questa valutazione tecnica si deve anche effettuare un'analisi di come venga percepito dai propri clienti il fatto di affidare a terzi l'esecuzione delle applicazioni e i dati ad esse relativi

Ad oggi non è ancora stato definito uno *standard condiviso* per l'utilizzo dei servizi di *Cloud Computing*, che favorirebbe sicuramente l'adozione di questo tipo di soluzioni da parte di potenziali clienti. Si ridurrebbero infatti gli effetti di *lock-in*, causati dalla complicazione delle procedure di migrazione da un provider all'altro. Uno dei più importanti tentativi di definire uno standard per i servizi di cloud computing è rappresentato da *Open Cloud Computing Interface (OCCI)* [51], che definisce i metodi per gestire e utilizzare le diverse risorse cloud, tuttavia molti tra i principali provider non hanno ancora accettato di utilizzare tali specifiche, mantenendo le proprie interfacce proprietarie, come Amazon API o VMware vCloud API. Molto probabilmente, con il diversificarsi e l'aumentare

dell'utenza, i provider si accorderanno sull'adozione di uno standard condiviso, garantendo così piena libertà agli utenti, che potranno scegliere il provider più adatto alle loro esigenze.

Inoltre bisogna sottolineare che per poter essere in grado sfruttare appieno tutte le funzionalità offerte dai cloud provider è necessario avere delle conoscenze sufficientemente avanzate in ambito sistemistico e di sviluppo software: utilizzare servizi IaaS richiede infatti di effettuare delle operazioni mirate per garantire il corretto funzionamento del sistema e delle applicazioni in esecuzione.

Un ultimo aspetto che potrebbe diventare molto importante nei prossimi anni è costituito dalla possibile interazione tra i servizi *cloud pubblici*, forniti dai service provider, e le risorse *private* presenti all'interno dei data center aziendali. Questo nuovo sistema di integrazione viene detto *Hybrid Cloud* [35]: un'azienda può utilizzare un software, come Eucalyptus, in grado di fornire servizi simili a quelli di un *cloud provider,* creando un servizio IaaS *privato.* Le risorse ottenute da questo sistema possono poi essere integrate, se necessario, con quelle offerte da un *cloud provider* pubblico. Ovviamente in questo caso è d'obbligo la presenza di un sistema in grado di interagire con la cloud "privata" e quella "pubblica", basato su standard condivisi. L'utilizzo di software di *private cloud* potrà quindi risultare molto utile in determinati scenari aziendali, consentendo di sviluppare sistemi che, di norma, utilizzano solamente le risorse interne, ma possono in ogni momento sfruttare quelle fornite da un *cloud provider* esterno, qualora fosse necessario.

12. BIBLIOGRAFIA

[1] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", *Future Generation Computer Systems*, vol. XXV, no. 6, pp. 599 - 616, 2009.

[2] L. Kleinrock, "A vision for the Internet", *ST Journal of Research*, pp. 4-5, 2005.

[3] Michael Armbrust et al., "Above the Clouds: A Berkeley View of Cloud Computing", EECS Department, University of California, Berkeley, Aprile 2009.

[4] Gartner, "Gartner Highlights Five Attributes of Cloud Computing", STAMFORD, June 23, 2009.

[5] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner, "A break in the clouds: towards a cloud definition", *SIGCOMM Comput. Commun. Rev.*, vol. XXXIX, pp. 50-55, Dicembre 2008.

[6] Amazon Web Services. [Online]. http://aws.amazon.com/

[7] Microsoft Azure. [Online]. http://www.microsoft.com/windowsazure/

[8] Google App Engine. [Online]. http://code.google.com/intl/it-IT/appengine/

[9] IBM Cloud. [Online]. http://www.ibm.com/ibm/cloud/

[10] Andrea Veraldi and Riccardo Chierici, "A quantitative comparison between xen and kvm", *Journal of Physics: Conference Series*, vol. CCXIX, no. 4, p. 042005, 2010.

[11] R. Buyya, Chee Shin Yeo, and S. Venugopal, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities", *High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference on*, pp. 5-13, Settembre 2008.

[12] Michael Miller. (2009, Febbraio) Cloud Computing Pros and Cons for End Users. [Online]. http://www.informit.com/articles/article.aspx?p=1324280

[13] James Staten, "Is Cloud Computing Ready For The Enterprise?", Forrester Research, 2008.

[14] Eucalyptus Community. [Online]. http://open.eucalyptus.com/

[15] Amazon Autoscaling. [Online]. http://aws.amazon.com/autoscaling/

[16] (2010) Defining Cloud Computing: Part 1-6. [Online]. http://clouddb.info/tag/definition/

[17] Apache Tomcat. [Online]. http://tomcat.apache.org/

[18] D. Patel Chandrakant and Shah Amip J, "Cost Model for Planning, Development and Operation of a Data Center", *HP Laboratories Palo Alto*, Giugno 2005.

[19] Guy Rosen, "The business of clouds", *Crossroads*, vol. XVI, no. 3, pp. 26-28, Marzo 2010.

[20] RightScale. (2010) Animoto's Facebook scale-up. [Online]. http://blog.rightscale.com/2008/04/23/animoto-facebook-scale-up/

[21] VMware, "Understanding Full Virtualization, Paravirtualization, and Hardware Assist", *Whitepaper*, Novembre 2007. [Online]. http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf

[22] T. Deshane et al., "Quantitative comparison of Xen and KVM", *Xen summit*, Giugno 2008.

[23] Johan De Gelas. (2008, Marzo) Hardware Virtualization: the Nuts and Bolts. [Online]. http://www.anandtech.com/show/2480/9

[24] CIO Weblog. (2008, Giugno) AppEngine outage. [Online]. http://www.cio-weblog.com/50226711/appengine_outage.php

[25] Allan Stern. (2008, Febbraio) Update From Amazon Regarding Friday's S3 Downtime. [Online]. http://www.centernetworks.com/amazon-s3-downtime-update

[26] The Health Insurance Portability and Accountability Act of 1996 (HIPAA) Privacy and Security Rules. http://www.hhs.gov/ocr/privacy/.

[27] Brian Krebs, "Amazon: Hey Spammers, Get Off My Cloud!", *Washington Post*, Luglio 2008.

[28] Amazon Web services. Washington Post Case Study. [Online]. http://aws.amazon.com/solutions/case-studies/washington-post/

[29] Apache. Welcome to Apache Hadoop! [Online]. http://hadoop.apache.org/

[30] OnLive. Welcome to OnLive.com. [Online]. http://www.onlive.com/

[31] Amazon AWS. SmugMug Case Study: Amazon Web Services. [Online]. http://aws.amazon.com/solutions/case-studies/smugmug/

[32] Don MacAskill. (2008, Giugno) SkyNet Lives! (aka EC2 @ SmugMug). [Online]. http://don.blogs.smugmug.com/2008/06/03/skynet-lives-aka-ec2-smugmug/

[33] Eric Y. Chen and Mistutaka Itoh, "Virtual smartphone over IP", *IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM)*, pp. 1-6, Giugno 2010.

[34] Markus Klems, Jens Nimis, and Stefan Tai, "Do Clouds Compute? A Framework for Estimating the Value of Cloud Computing", *Designing E-Business Systems. Markets, Services, and Networks*, vol. XXII, pp. 110-123, 2009.

[35] Philipp C. Heckel, "Hybrid Clouds: Comparing Cloud Toolkits", *Seminar Paper, University of Mannheim*, Aprile 2010.

[36] A. Khajeh-Hosseini, D. Greenwood, and I. Sommerville, "Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS", *IEEE 3rd International Conference on Cloud Computing (CLOUD)*, pp. 450 -457, Luglio 2010.

[37] VMware. VMware vCloud™ Express. [Online]. http://www.vmware.com/solutions/cloud-computing/public-cloud/vcloud-express.html

[38] Bluelock. Bluelock Cloud Hosting. [Online]. http://www.bluelock.com/bluelock-cloud-hosting/

[39] Hosting.com Cloud. [Online]. http://www.hosting.com/

[40] Terremark. Terremark Worldwide Inc. [Online]. http://www.terremark.com/default.aspx

[41] Seeweb. (2010, Settembre) Seeweb Cloud Hosting. [Online]. http://www.seeweb.it/cloudhosting/

[42] GoGrid. Cloud Hosting, Cloud Servers, Hybrid Hosting, Cloud Infrastructure from GoGrid. [Online]. http://www.gogrid.com/

[43] Rackspace. Rackspace hosting. [Online]. http://www.rackspace.com

[44] Oracle. Amazon Web Services and Oracle Announce Certification and Support of Oracle Applications, Middleware and Databases on Amazon Elastic Compute Cloud using Oracle VM. [Online]. http://www.oracle.com/us/corporate/press/173480

[45] Gerardo Viedema. (2010, Luglio) Persistence Strategies for Amazon EC2. [Online]. http://www.theserverlabs.com/blog/2010/07/08/ec2-persistence-strategies/

[46] Sean A. Walberg, "Migrate your Linux application to the Amazon cloud, Part 2: Improving application reliability", Agosto 2010. [Online]. Migrate your Linux application to the Amazon cloud, Part 2: Improving application reliability

[47] Daniel Nurmi et al., "The Eucalyptus Open-Source Cloud-Computing System", *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp. 124-131, 2009.

[48] Eucalyptus. Eucalyptus Enterprise Edition. [Online]. http://www.eucalyptus.com/

[49] Openstack. OpenStack Open Source Cloud Computing Software. [Online]. http://www.openstack.org/

[50] OpenNebula: The Open Source Toolkit for Cloud Computing . [Online]. http://www.opennebula.org/

[51] OGF Open Cloud Computing Interface Working Group. [Online]. http://www.occi-wg.org

[52] Reservoir fp7. [Online]. http://62.149.240.97/

[53] Intalio, Inc. - Intalio The Private Cloud Company. [Online]. http://www.intalio.com/

[54] VMware vSphere 4: Private Cloud Computing, Server and Data Center Virtualization. [Online]. http://www.vmware.com/products/vsphere/

[55] VMWare. VMware vSphere 4 - Datasheet. [Online]. http://download3.vmware.com/media/press/VMware-vSphere-4.0-Brochure-Data-Sheet.pdf

[56] RightScale. (2010, Aprile) Benchmarking Load Balancers in the Cloud. [Online]. http://blog.rightscale.com/2010/04/01/benchmarking-load-balancers-in-the-cloud/

[57] Amazon. (2010, Aprile) New Elastic Load Balancing Feature: Sticky Sessions. [Online]. http://aws.typepad.com/aws/2010/04/new-elastic-load-balancing-feature-sticky-sessions.html

[58] Shlomo Swidler. (2010, Aprile) Apache Load Balance Using Haproxy - sticky sessions. [Online]. http://shlomoswidler.com/2010/04/elastic-load-balancing-with-sticky-sessions.html

[59] Amazon AWS. Amazon Auto Scaling - Developer Guide. [Online]. http://docs.amazonwebservices.com/AutoScaling/latest/DeveloperGuide/

[60] Harish Ganesan. Cloud Computing : Autoscaling and Elastic Load balancing using Amazon AWS. [Online]. http://www.slideshare.net/harishganesan/cloud-computing-autoscaling-and-elastic-load-balancing-using-amazon-aws

[61] Amazon. Using Parameterized Launches to Customize Your AMIs. [Online].

http://aws.amazon.com/articles/1085?_encoding=UTF8&jiveRedirect=1

[62] SYSSTAT. [Online]. http://sebastien.godard.pagesperso-orange.fr/

[63] HAProxy. The Reliable, High Performance TCP/HTTP Load Balancer. [Online]. http://haproxy.1wt.eu/

[64] HAProxy. New benchmark of HAProxy at 10 Gbps using Myricom's 10GbE NICs (Myri-10G PCI-Express). [Online]. http://haproxy.1wt.eu/10g.html

[65] HATop. hatop - Project Hosting on Google Code. [Online]. http://code.google.com/p/hatop/

[66] George Reese. (2009, Marzo) The Weakness of Commodity Server to Cloud Server Cost Comparisons. [Online]. http://broadcast.oreilly.com/2009/03/comparing-cloud-costs.html