ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

FACOLTÀ DI INGEGNERIA
CORSO DI LAUREA IN INGEGNERIA INFORMATICA

ANALISI E SVILUPPO DI STRUMENTI PERFORMANCE-AWARE PER LA MIGRAZIONE CLOUD ENTERPRISE

Tesi in
SISTEMI DISTRIBUITI L-S

Relatore:
Chiar.mo Prof. Ing.
PAOLO BELLAVISTA

Correlatore:
Prof. Ing.
ANTONIO CORRADI

Correlatore:
Dott.
EMILIO LUCOTTI

Presentata da:
ANDREA DECORTE

Sessione III
Anno Accademico 2010 – 2011

Dedicât a la nona, che prea simpri par me

Indice

Elenco delle figure

Introduzione

Il tema del cloud computing è già da tempo al centro delle discussioni nel mondo dell'Information Technology. I vantaggi derivanti da questa evoluzione tecnologica e il buon livello di maturità raggiunto stanno rapidamente attirando l'attenzione su questo argomento anche da parte delle aziende. Esse, d'altra parte, hanno dei requisiti piuttosto stringenti, per esempio riguardo alla protezione dei dati e, prima di ogni evoluzione significativa, devono affrontare il tema della migrazione di architetture legacy. Queste problematiche impongono un'attività preliminare di analisi, che potrebbe essere piuttosto onerosa dal punto di vista dei tempi e dei costi: da qui nasce l'esigenza di strumenti e procedure automatizzate, che, in questa fase, vengano in aiuto dell'analista e possano dare una prima idea delle possibilità di migrazione. Per la variabilità intrinseca di un ambiente cloud non sarà possibile raggiungere una grande precisione, ma i risultati verranno poi confermati direttamente al momento dell'esecuzione; le configurazioni non saranno mai fissate come in un ambiente tradizionale non virtualizzato, ma si potrà intervernire in ogni momento.

La progettazione e implementazione di questo tipo di strumenti sono al centro della tesi, che si basa, inoltre, sulle esperienze e i casi affrontati durante l'internship presso IBM. Il lavoro si concentra, in particolare, sull'aspetto infrastrutturale, prendendo in considerazione i sistemi operativi e i middleware che essi ospitano. Come ambiente di destinazione viene prevista la possibilità di avere sia ambienti cloud pubblici che privati, con le loro peculiarità, restringendo l'analisi a offerte che mettono a disposizione solamente il sistema operativo, senza software aggiuntivo. Lo studio si orienta in due direzioni principali: la prima parte dai dati raccolti sulle configurazioni dei sistemi esistenti, che con degli appositi parametri vengono confrontati con le possibili configurazioni di destinazione, in modo da arrivare alla miglior corrispondenza e scartando i sistemi per i quali la migrazione non è possibile. I dati raccolti verranno inseriti in un database relazionale, sul quale l'utente

può agire sia tramite gli algoritmi sviluppati nel corso del lavoro, sia in maniera personalizzata. Quest'ultimo vincolo, insieme alla modularizzazione della soluzione sviluppata, è necessario per mantenere quella flessibilità necessaria ad affrontare un ambiente così vario e sfaccettato, nell'ottica anche di una possibile integrazione con altre soluzioni IBM. Il sistema è stato messo alla prova su diversi casi di studio reali, ottenendo risultati corretti e soddisfacenti dal punto di vista dell'utente finale.

Nella seconda parte si andrà a realizzare una serie di procedimenti originali, il più possibile automatizzati, in grado di analizzare le prestazioni reali di un sistema migrato, così da individuare la configurazione sull'ambiente cloud di destinazione che sia in grado di offrire il miglior compromesso possibile tra costi e prestazioni possibili. Come si è detto, non sarà necessario effettuare test molto dettagliati né prendere in considerazione l'ipotesi di picchi, poiché tali situazioni si possono gestire in modo dinamico. Viene inoltre prevista, in entrambi gli scenari, la possibilità di considerare diversi provider di servizi cloud contemporaneamente, ampliando così le possibilità di scelta. Lo scenario dinamico è stato messo alla prova su uno scenario di una tipica *web application,* con considerazioni sulle modalità di scalabilità possibili e, in particolare, su quella orizzontale, che si presta molto ad un'ambiente di tipo cloud.

Nel capitolo 1 presentiamo una introduzione generale al tema del cloud computing, cercando prima di ottenere una definizione e poi studiando le varie suddivisioni proposte, concludendo con l'importante tema dei vantaggi e dei problemi ancora aperti. Nel capitolo 2 si descrive un'architettura generale di riferimento per il cloud proposta da IBM, andando poi nei dettagli per quanto riguarda l'offerta di cloud pubblico dell'azienda, sulla quale verranno svolte le attività implementative e i test. Nel capitolo 3 introduciamo l'argomento principale di questa tesi, l'analisi per la migrazione verso un ambiente cloud, con particolare attenzione all'ambito enterprise e fotografiamo lo stato dell'arte. Nel capitolo 4 viene affrontata la fase di progetto degli strumenti e delle procedure sviluppate per quanto riguarda i due scenari visti, quello statico e quello dinamico. Nel capitolo 5, infine, si descrive l'implementazione e i risultati dei test effettuati, dando maggior spazio alla correttezza dei risultati per l'analisi di tipo statico e alle prestazioni nello scenario dinamico; viene inoltre trattato, da un punto di vista pratico, il tema dell'utilizzo di provider differenti, che apre la strada per scenari di gestione dei picchi di richieste.

Capitolo 1

Cloud computing

Il cloud computing è una nuova *information technology wave* in grado di far evolvere gli attuali sistemi informativi attraverso nuove modalità di erogazione di servizi informatici. Questo cambiamento, facilitato, fra le altre cose, dal continuo calo del costo delle risorse computazionali fisiche, si basa sulle precedenti *wave*, come viene riassunto da Gartner Group:

> *Cloud is emerging at the convergence of three major trends: service orientation, virtualization and standardization of computing through the Internet.*

Il concetto di cloud e, in particolare, quello del computing come facility, si riscontra già negli anni '60 [Par66], ma è solo nel 2006 che sale alla ribalta della cronaca dopo un discorso di Eric Schmidt, CEO di Google, che utilizzò tale parola per descrivere il modello di business della fornitura di servizi per mezzo della rete Internet [SS06]. Ancora oggi, tuttavia, le definizioni di cloud sono molto varie, il che genera confusione e scetticismo; tra le più citate, troviamo quella del National Institute of Standards and Technology (NIST):

> *Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction*[MG11].

Questa definizione conferma che non ci troviamo ad affrontare una nuova tecnologia, ma un nuovo modello, nato per supportare esigenze di business nuove e preesistenti, in grado

di coniugare soluzioni tecnologiche mature come la virtualizzazione o innovative come la capacità di misurare e addebitare i consumi effettivi delle risorse.

Risulta interessante notare come il nuovo paradigma del cloud computing abbia portato anche a una suddivisione di ruoli: il classico ruolo del service provider si articola ora in quello dell'infrastructure provider, che gestisce la parte fisica del sistema e noleggia le risorse fisiche, in genere con modelli di tipo "utility" e i service provider veri e propri, che sfruttano le risorse prese a noleggio per fornire dei servizi e possono quindi guadagnare in flessibilità e riduzione dei costi.

Oltre a quella appena citata, sono state proposte altre definizioni: il laboratorio RAD dell'Università di Berkeley definisce il cloud computing in modo molto più ristretto come la somma di un pricing in stile utility e di Software-as-a-Service. Tutte le altre tipologie che analizzeremo tra poco e l'intero strato hardware vengono inclusi in questa visione sotto la definizione più generale di cloud [AFG+10]. In [VRMCL08] troviamo invece la raccolta di più di venti definizioni diverse e a volte abbastanza discordanti, dalle quali gli autori cercano di distillarne una con tutti gli elementi di base del cloud computing: tale definizione, orientata in particolare verso l'aspetto infrastrutturale del cloud, pone l'accento su scalabilità, pricing in base all'utilizzo e virtualizzazione come elementi unificanti.

1.1 Tecnologie di base

Alla base del cloud computing troviamo una serie di tecnologie, che a volte vengono confuse con il cloud stesso; vediamo di seguito di chiarirne similitudini e differenze per evidenziare come il Cloud si basi sull'armonizzazione di tecnologie preesistenti.

L'architettura orientata ai servizi, o Service Oriented Architecture (SOA), indica generalmente un'architettura software adatta a supportare l'uso di servizi Web per garantire l'interoperabilità tra diversi sistemi, in modo da consentire l'utilizzo delle singole applicazioni come componenti del processo di business e soddisfare le richieste degli utenti in modo integrato e trasparente. SOA e cloud sono per certi aspetti in sovrapposizione; le implementazioni di SOA, tuttavia, sono fondamentalmente tecnologie di integrazione applicativa che abilitano lo scambio di informazioni tra sistemi informativi enterprise, permettendo inoltre di avere uno strato comune di invocazione indipendente dai singoli linguaggi. Il cloud invece ha un obiettivo di più ampio respiro, cercando di sfruttare la rete per effettuare un outsourcing di tutti i componenti tecnologici, dall'hardware fino al software.

Ciò va quindi molto al di là degli obiettivi di SOA e queste due metodologie possono essere considerate complementari e procedere di pari passo all'interno di un'azienda[Rai09]. Soluzioni applicative basate su SOA si sposano molto bene con un'architettura cloud basata su servizi a catalogo e possono essere riutilizzate all'interno di essa.

Il grid computing è un paradigma di computing distribuito che coordina risorse in rete per ottenere determinati risultati in un tempo minore di quello che sarebbe richiesto in caso di esecuzione su una singola macchina; l'idea alla base è quella di dividere un compito in parti da riunire poi, al termine, su un nodo centrale. C'è spesso una confusione tra cloud e grid computing: in generale possiamo affermare che il cloud computing può sostenere i compiti di una soluzione grid, poiché è possibile distribuire la potenza computazionale nel cloud, mentre non è vero il contrario; il cloud inoltre non richiede necessariamente applicazioni specifiche, ma può ospitare anche applicazioni tradizionali. In [VRMCL08] gli autori sostengono che, anche se probabilmente si andrà verso una convergenza dei due campi, il cloud computing si differenzia per avere alla base la virtualizzazione come concetto fondante, per una maggiore attenzione per l'usabilità oltre che per sicurezza e Quality of Service (QoS); anche il concetto di utility computing di solito non è presente nel grid, dove il pagamento è fisso. Si sostiene, infine, come il cloud dovrebbe cercare di ispirarsi al grid computing per quanto riguarda la standardizzazione, per la quale in quest'ultimo paradigma sono stati intrapresi ingenti sforzi.

L'utility computing è l'idea di far pagare le macchine in base all'utilizzo, come avviene per l'acqua o per la corrente elettrica (le cosiddette *commodity*); questa idea trova un'applicazione completa nel cloud a livello infrastrutturale, mentre la sua applicazione a livelli superiori, in particolare per quanto riguarda il software, è ancora in fase di evoluzione. Attraverso l'utility computing, che viene spesso definito in termini pratici come *pay-as-you-go*, l'utente è in grado di sapere in ogni momento il costo di utilizzo e quindi di poter offrire risparmi anche nel caso di risorse utilizzate solamente in determinati orari (ad esempio per macchine che devono rimanere accese solo nei giorni feriali oppure che servono per momenti di traffico intenso).

La virtualizzazione è una tecnologia presente fin dagli anni '60 sulle piattaforme mainframe; negli ultimi anni l'evoluzione tecnologica ne ha provocato una grande diffusione in ambito aziendale, al fine di realizzare ottimizzazioni sia delle componenti dei datacenter quali server, storage e rete che di quelle distribuite come i desktop. Essa permette l'astrazione dai dettagli delle macchine fisiche, consentendo un migliore sfruttamento delle

risorse – in uno scenario tradizionale le macchine rimangono spesso inattive per lunghi periodi, per cui gran parte della capacità computazionale non viene usata, mentre la virtualizzazione consente ridistribuzioni dinamiche – e di offrire risorse virtualizzate per applicazioni di alto livello; è anche uno dei mattoni fondanti dello strato hardware del cloud computing, permettendo la rapida creazione di istanze e la gestione del carico in caso di picchi. Un ambiente già virtualizzato facilita la migrazione verso il cloud e la migrazione ad un ambiente virtualizzato, per quanto riguarda le problematiche da affrontare, si avvicina significativamente a quella verso un ambiente di tipo cloud.

L'autonomic computing, infine, è un termine lanciato nel 2001 dall'IBM per indicare sistemi computazionali completi in grado di autogestirsi, reagendo a modifiche e stimoli esterni senza interventi da parte di un operatore. L'obiettivo è quello di ridurre la complessità di gestione anche per quanto riguarda ottimizzazioni e sicurezza, che è in continuo aumento nei sistemi moderni. Tale idea viene parzialmente ripresa nel cloud computing, per esempio per quanto riguarda il provisioning automatico di risorse, ma l'accento viene spesso posto più sulla riduzione dei costi che sulla gestione automatica.

1.2 Caratteristiche principali

Tra le caratteristiche alla base del cloud troviamo:

1. On-demand self-service: i servizi sono disponibili su richiesta in un periodo molto breve (nell'ordine di ore piuttosto che di giorni), senza necessità di dover contattare il fornitore

2. Broad network access: l'accesso alla rete deve ovviamente essere ampio e affidabile e solo negli ultimi anni i collegamenti moderni ad alta banda hanno reso possibile l'utilizzo di servizi e di software che non siano localizzati esclusivamente vicino al punto di utilizzo, ma potenzialmente in ogni parte del globo

3. Resource pooling: le risorse hardware vengono condivise in modo trasparente all'utente finale. Può essere associata alla location independence, cioè alla condivisione di risorse anche in luoghi diversi: questo può essere un ottimo accorgimento per ottenere un'alta availability, ma d'altra parte non consente di conoscere precisamente dove siano i dati. Dal punto di vista aziendale, inoltre, il resource pooling rende

possibile affrontare il tema del *capacity planning* a un livello globale di centro di elaborazione dei dati anziché per ogni singola applicazione o soluzione di business

4. Rapid elasticity: anche se a lungo andare i costi di acquisto di un server potrebbero essere inferiori a quelli del noleggio di risorse in cloud, l'elasticità permette di avere sempre un utilizzo pari alle necessità; risulta infatti come i server di Google siano in genere utilizzati tra il 10% e il 50%[Vog08], proprio perché sono dimensionati per i picchi. Questo spreco di risorse può in molti casi controbilanciare eventuali costi superiori di una soluzione in cloud, nelle quali possiamo reagire in pochi minuti a picchi di richieste

5. Illusion of unlimited hardware: grazie alla scalabilità automatica, l'utente non deve preoccuparsi delle risorse hardware a sua disposizione, ma può illudersi di avere a disposizione una potenzialità computazionale infinita e disponibile in pochi minuti

6. Measured Service: il servizio si paga in base all'utilizzo e può essere disattivato in qualsiasi momento senza ulteriori aggravi economici. In [YBDS08] si distingue tra tre modalità di pagamento emerse per i servizi legati al cloud computing: la prima è quella del *tiered pricing*, in cui i servizi possono essere scelti tra una serie di opzioni predefinite (per esempio istanze dotate di CPU, memoria e spazio di storage fisso) che hanno un determinato prezzo per unità di tempo. Troviamo poi il *per-unit pricing*, maggiormente flessibile e normalmente applicata ai trasferimenti di dati o all'utilizzo di spazi di memorizzazione, in cui si paga la quantità utilizzata per unità di tempo. Infine esiste anche un modello *subscription based*, che richiede un prezzo prefissato per l'utilizzo e gode di diffusione in particolare per il Software-as-a-Service; quest'ultimo modello consente di predire con precisione i costi, ma non permette un'accurata misurazione dei consumi effettivi.

1.3 Architettura del cloud computing

L'architettura del cloud può essere suddivisa, in base al loro compito, in quattro livelli: il livello del datacenter, quello dell'infrastruttura, quello della piattaforma e quello del software. Di questi, quelli visibili e controllabili da parte dell'utente finale, privato o azienda

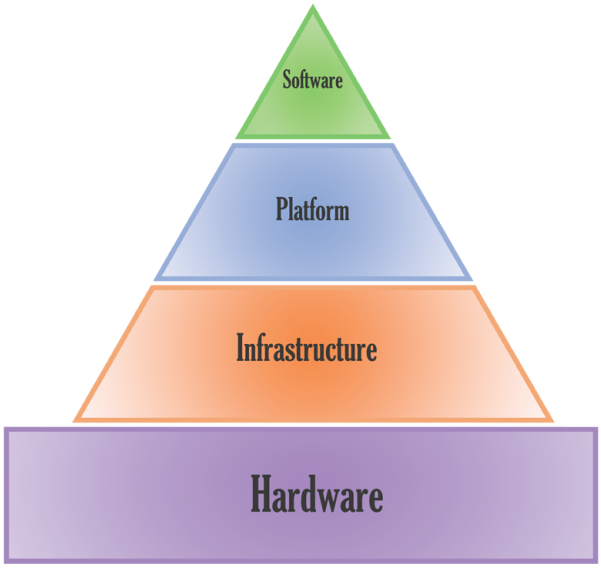

Figura 1.1: I livelli del cloud computing

che siano, sono gli ultimi tre. La visione IBM si distanzia inoltre da questo modello perché prevede, al di sopra del livello del software, un ulteriore livello, quello dei Business Process, che si propone di spostare la disciplina del *business process management* in uno scenario cloud.

Nel caso dei maggiori cloud service provider come IBM o Amazon, si può considerare la presenza di un ulteriore livello logico, quello Enterprise; in tal caso, infatti, i sistemi informativi si basano su più datacenter cloud distribuiti a livello globale, ognuno dei quali implementa i quattro livelli di architettura descritti.

Possiamo notare come i livelli siano componibili ma indipendenti e possano quindi potenzialmente essere serviti da provider diversi: se, nella maggior parte dei casi, i livelli IaaS e PaaS sono forniti dallo stessa azienda, per quanto riguarda il Software-as-a-Service sempre più produttori propongono i loro prodotti su sistemi di provider differenti.

1.3.1 Hardware layer

Al livello più basso, i sistemi di cloud computing si basano su datacenter, di dimensioni in genere elevate per sfruttare l'economia di scala, sui server dei quali vengono create e distrutte le diverse macchine virtuali. È interessante notare come la maggior parte dei for-

nitori cloud propendano per la creazione di datacenter nuovi, separati da quelli tradizionali; tale separazione è valida sia dal punto di vista delle componenti tecnologiche utilizzate che delle strutture organizzative.

Le problematiche di questo livello vengono completamente nascoste all'utente finale, al quale viene presentata solo la visione di macchine virtuali di taglie diverse, a volte senza neanche specificare l'esatta locazione fisica nella quale la VM verrà posizionata. La loro realizzazione, tuttavia, pone al cloud provider dei problemi tecnici non indifferenti sia progettuali che realizzativi. In Barroso e Hölzl [BH09] si sostiene come per il cloud computing si assista ad un'evoluzione dei datacenter, che porta gli autori a coniare il termine di *warehouse-scale computers*: in questo nuovo scenario, il datacenter appartiene a una singola organizzazione, l'hardware e il software sono omogenei e c'è uno strato comune di amministrazione. Anche da un punto di vista esterno la visione che si ottiene è quella di una singola unità computazionale più che un semplice aggregato di macchine. Un'altra novità di questa visione è che, a causa del numero elevato di componenti, ce ne saranno sempre di non funzionanti, per cui l'idea è di cercare di gestire lo spostamento di carichi elaborativi su altri componenti funzionanti mediante tecniche applicative o componenti di middleware, così da evitare che l'utente ne debba essere a conoscenza. La programmazione di questi sistemi è complessa anche a livello di rete e storage, che sono i colli di bottiglia di questi sistemi, quindi la ricerca si orienta verso l'uso di file system distribuiti globali come Google's File System (GFS)[GGL03]. L'ultima parte affrontata dagli autori è quella dell'efficienza, che è in continuo aumento, ma rimane uno dei costi maggiori di questi sistemi, anche perché la maggior parte dei componenti informatici non presentano significativi risparmi energetici se lavorano al di sotto del massimo possibile, che è la condizione più comune in datacenter di queste dimensioni.

1.3.2 Infrastracture layer (IaaS)

In [YBDS08], che è uno dei primi studi su come suddividere e classificare il cloud computing, lo IaaS viene definito come lo strato che fornisce risorse fondamentali per gli strati superiori e, a sua volta, può essere utilizzato per costruire nuovi ambienti software o applicazioni in cloud o applicazioni. La definizione del NIST invece è:

> *The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer*

> *is able to deploy and run arbitrary software, which can include operating systems and applications*[MG11].

Il livello IaaS raccoglie le offerte di più basso livello per l'utente finale: vengono fornite le risorse computazionali di base quali capacità di calcolo, di memorizzazione dati e comunicazione in rete, offerte in genere sotto forma di Virtual Machine chiamate *istanze*. IBM copre questo segmento con la sua offerta Smart Cloud Enterprise[IBM], mentre Amazon offre la piattaforma Elastic Cloud (EC2) [Ama12]. Va notato come sia più difficile offrire a questo livello strumenti automatici di scalabilità e failover, in quanto la semantica della replicazione e di altri strumenti di management sono application-dependent, mentre essi sono fornibili in modo molto più semplice a un livello PaaS, come si vede dall'offerta di Google App Engine. In [YBDS08] lo strato viene ulteriormente scomposto in *computational resources*, *data storage* (a volte denominato Database-as-a-Service) e *communications*; quest'ultima parte è quella che ha avuto meno sviluppi pratici e, secondo gli autori, ha lo scopo di fornire un servizio di comunicazione orientato ai servizi, configurabile, programmabile, prevedibile e affidabile.

1.3.3 Platform layer (PaaS)

La definizione del NIST è:

> *The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider*[MG11].

In questo caso, rispetto al livello precedente, il provider fornisce uno strato aggiuntivo di tipo software che facilita l'utilizzo delle caratteristiche più importanti del cloud, fra le quali l'elasticità. Tra i più famosi provider in questo settore troviamo Google con il suo App Engine [Goo], basato su una tecnica di sandbox, Amazon e Microsoft Windows Azure [Mic12].

Si può notare come all'interno di questa fascia si possano avere diversi approcci: quello di Google, per esempio, fornisce un framework completo destinato esclusivamente ad applicazioni Web tradizionali (organizzate secondo l'architettura a tre tier) e impone una rigida separazione tra il tier computazionale stateless e quello di storage che invece è stateful. I linguaggi disponibili sono Java e Python e viene supportato solamente un database di

tipo non relazionale, una tipologia che comunque è in notevole espansione, mentre l'accesso al filesystem è esclusivamente in lettura. Altre limitazioni sono quelle di poter utilizzare solamente un sottoinsieme delle librerie Java Standard Edition e l'impossibilità di lanciare thread, laddove dal lato Python non si possono usare moduli ed estensioni non Python (per esempio quelle scritte in linguaggio C). È disponibile per questa offerta anche un IDE che ci permette di effettuare il testing in locale ed effettuare poi la pubblicazione online in modo semplice. Questa proposta, dunque, sembra essere particolarmente orientata per applicativi Web che vogliono sfruttare le doti di scalabilità permesse dal cloud senza necessità di configurazioni aggiuntive e per i quali le limitazioni presenti non sono un ostacolo.

Azure invece è più flessibile come applicazioni supportate, fornisce un supporto anche per database relazionali classici e, sfruttando la base del .NET framework, permette di utilizzare un buon numero di linguaggi di programmazione; tuttavia, il supporto automaticamente fornito per failover e scalabilità è minore e quindi, nel caso della migrazione di un'applicazione Web classica a tre tier, non sembra essere la scelta migliore, a meno di utilizzare già la tecnologia .NET oppure un linguaggio non supportato dalla soluzione di Google come PHP.

1.3.4 Software layer (SaaS)

> *The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure*[MG11].

Questo strato fa riferimento alla fornitura di software come servizi accedibili, senza necessità di installazioni sulla singola macchina ed è in genere destinato agli end user. Sta avendo una notevole diffusione in alcuni settori, per esempio quello dell'email e offre una serie di vantaggi sia da parte del provider (riduzione di problemi di gestione e di aggiornamento, visto che tutti useranno sempre l'ultima versione), sia dal punto di vista dell'utente (possibilità di accedere da qualsiasi dispositivo connesso avendo a disposizione sempre lo stesso ambiente di lavoro e tutti i dati, facilità di condivisione, ecc.). Fra i punti critici aperti, quello del licensing, l'obbligo di utilizzare solo la versione più aggiornata e la necessità di avere garanzie sull'availability del servizio, in quanto un blocco non è facilmente risolvibile da parte dell'utente. Importante è anche la possibilità nel caso di cambio di provider di esportare i dati verso formati comuni per evitare il lock-in, problematica che affronteremo in dettaglio fra poco.

1.3.5 Business Process layer (BPaaS)

L'IBM definisce tale strato come:

> *Business process services are any business process (horizontal or vertical) delivered through the Cloud service model (Multi-tenant, self-service provisioning, elastic scaling and usage metering or pricing) via the Internet with access via Web-centric interfaces and exploiting Web-oriented cloud architecture. The BPaaS provider is responsible for the related business function(s)*[BPa10].

A volte conosciuto con altri nomi, ad esempio quello di Enterprise-as-a-Service, questo livello si inserisce all'interno della tematica più ampia del *business process management*, che è un approccio strutturato basato su metodi, politiche, metriche e *practice* di management, per gestire ed ottimizzare continuamente le attività e i processi di un'organizzazione. Questo avviene per mezzo di un layer intermedio in grado di consentire una suddivisione netta tra le regole e la logica applicativa, vale a dire un motore di esecuzione; per questo motivo, può essere visto come un'estensione del livello precedente, in quanto è in pratica un software spinto fino a livelli massimi di configurabilità.

Tutto questo in una visione cloud può essere spostato al di fuori del controllo dell'utente, che potrà accedere ai servizi attraverso un'interfaccia Web e tramite architetture Web oriented.

Al momento, secondo la visione IBM, si può sfruttare il prodotto Blueworks Live per spostare in cloud i processi di business meno complessi e, in generale, la fase di creazione e di aggiornamento delle regole, che richiede un test approfondito per analizzare le conseguenze; questa attività richiederebbe delle macchine di sviluppo, i cui costi fissi possono essere evitati sfruttando la scalabilità dell'infrastruttura cloud.

Un esempio di possibile target di questo layer è una banca, che può modellizzare e migliorare continuamente tutti i suoi processi, evitando i costi fissi della fase di test e sviluppo e dovendo solo investire nei server che gestiscono i motori di esecuzione delle varie regole. Oltre a IBM, altre aziende propongono soluzioni di questo tipo e l'azienda di ricerca Forrester predice dei forti sviluppi per questo segmento di mercato, che potrà raggiungere un valore di 10 miliardi di dollari nel 2020.

1.4 Modelli di deployment di cloud

Se la visione precedente offriva una prospettiva dal lato dell'utilizzatore, vediamo ora le distinzioni che si possono fare dal punto di vista del provider di cloud computing, partendo dai quattro modelli presi in considerazione nella definizione del NIST.

1.4.1 Public cloud

In questo scenario, il provider offre il suo servizio a un pubblico generico per cui le risorse sono condivise tra tutti gli utenti, anche se l'utilizzatore ritiene di essere l'unico a usufruirne; tale caratteristica viene in genere definita come *multi-tenancy*. La connessione può essere normale oppure si può anche sfruttare una VPN per collegarsi direttamente con la rete aziendale; è la soluzione più comune e in questo settore vi è una forte concorrenza, con la presenza di tutte le grosse aziende e di molte medio-piccole.

1.4.2 Private cloud

Si indicano con questo termine cloud progettati per l'utilizzo di una singola azienda, per cui non c'è alcuna condivisione con altri utenti; possono essere completamente gestiti dall'utente stesso oppure ospitati presso il datacenter del provider del servizio. Pur fornendo vantaggi quali l'elasticità e la possibilità di creare istanze in tempi brevissimi, la loro creazione richiede un investimento iniziale per cui si avvicina notevolmente a una soluzione di tipo outsourcing presente sul mercato da diversi anni; come nel caso dell'outsourcing, è possibile garantire un livello di sicurezza e resilienza molto più elevato, visto che possono essere introdotti maggiori controlli su accessi e connessioni di rete dedicate.

1.4.3 Hybrid cloud

È una soluzione intermedia che cerca di riunire i vantaggi dei due casi precedenti ed è tra le soluzioni a maggior crescita, in quanto risponde meglio alle esigenze delle aziende che possono, mediante questo modello, usufruire dei servizi Cloud in modo graduale, sostituendoli ad alcuni degli attuali servizi applicativi e integrandoli con i rimanenti. In questo caso i servizi informatici del cliente (IaaS, PaaS, ecc.) sono erogati da un'infrastruttura distribuita tra i datacenter del cliente e del provider. In particolare, nel caso di servizi di

tipo IaaS, per il cliente si riducono i problemi di dimensionamento e gestione di un'infrastruttura privata; la tematica principale di un hybrid cloud è capire come distribuire nel modo migliore i carichi elaborativi (workload) fra private e public e come integrare le due architetture per mezzo di tecnologie standard o proprietarie senza creare colli di bottiglia.

Una soluzione che può essere inserita in questa categoria è la *Virtual private cloud*, che, pur sfruttando l'infrastruttura del public cloud, viene dotata di topologia di rete personalizzabile e impostazioni aggiuntive di sicurezza, rendendola quindi meno esposta ai rischi esterni senza necessità di pagare i costi di creazione di un cloud privato[ZCB10]; proposte commerciali di questo tipo sono previste sia da Amazon a livello di infrastruttura[1] sia da Google a livello di piattaforma[2].

1.4.4 Community cloud

In questo caso l'infrastruttura di tipo cloud viene creata esclusivamente per l'uso da parte di una comunità specifica che ha preoccupazioni condivise riguardo a temi come la sicurezza, le politiche di accesso e così via. L'infrastruttura può essere gestita sia da uno degli utilizzatori che da terze parti; in questo caso si suddividono i costi, rispetto ad uno scenario di private cloud, ma si riducono ad ogni modo i vantaggi derivanti dall'adozione del nuovo paradigma.

1.4.5 La visione IBM

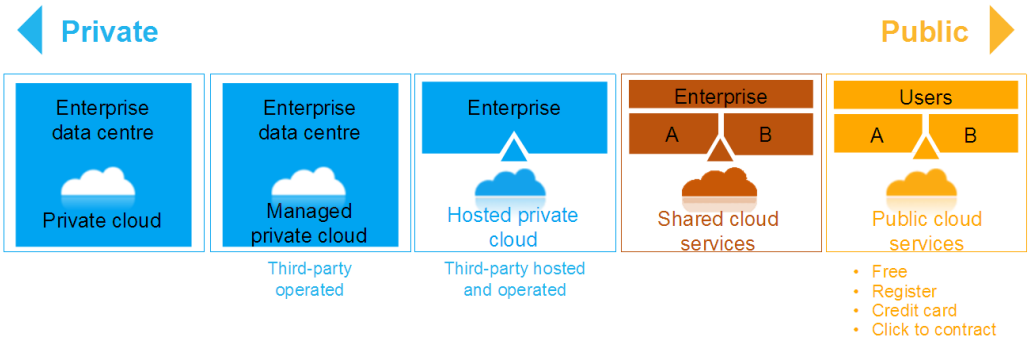

Figura 1.2: I modelli di deployment in una visione IBM

[1] http://aws.amazon.com/vpc/
[2] http://code.google.com/securedataconnector/

Analizziamo adesso, dopo questa panoramica, la categorizzazione dei modelli di deployment di figura 1.2, presentata dall'IBM, nella quale viene posta particolare attenzione all'ambito enterprise. I due estremi di questa visione sono il public cloud da una parte e il private dall'altra, mentre, al suo interno, ci sono delle sottocategorie che vengono trascurate nella visione più generale del NIST.

Partendo da destra, la visione più restrittiva, ma che garantisce maggior controllo e dunque sicurezza, è quella di un cloud privato costruito nel datacenter dell'azienda; esso poi può anche essere gestito da una terza parte, pur rimanendo sotto il controllo dell'azienda, con i conseguenti vantaggi per la protezione dei dati sensibili. Lo scenario previsto al centro è quello di un cloud ospitato e gestito da una terza parte, mentre, spostandoci verso destra, si va verso uno scenario di tipo public, nel quale i servizi sono condivisi, ma il target è quello enterprise (come vedremo più tardi è il segmento coperto da IBM Smart Cloud Enterprise) e infine verso uno scenario completamente public, in cui chiunque può registrarsi e pagare attraverso carta di credito o mezzi similari (è lo scenario coperto, per esempio, dalla soluzione Amazon EC2); in quest'ultimo caso, ovviamente, i costi saranno molto più bassi, ma lo sarà anche la possibilità di controllo sulla piattaforma.

1.5 Vantaggi

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

1. Nessun up-front investment: non vi è più la necessità di grandi investimenti iniziali, perché il pagamento avviene in base all'utilizzo. Questo permette un notevole risparmio soprattutto nel caso di un ambiente di test e sviluppo, dove le macchine servono per un periodo di tempo limitato e rimarrebbero poi inutilizzate

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

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

4. Riduzione dei rischi di impresa e dei costi operativi e di manutenzione: come per i servizi di outsourcing, modello al quale spesso il cloud si avvicina molto, l'azienda, delegando i suoi sistemi a una entità esterna specializzata in grado di fornire un servizio a costi inferiori, può ridurre i propri costi; si facilita inoltre una riorganizzazione interna, consentendo lo spostamento di esperti IT dai compiti di più basso livello quale la riparazione di server aziendali, verso attività di innovazione e ottimizzazione delle soluzioni esistenti.

1.6 Problemi aperti

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

1.6.1 Connettività

Il cloud computing, per la sua stessa natura di servizio usufruibile attraverso una rete accessibile da chiunque in qualunque posto, è nato grazie alla rete Internet e dipende dalla capacità di Internet di offrire una connessione efficiente e affidabile; un problema di connettività può provocare un blocco completo del processo produttivo legato alla disponibilità dei sistemi informativi cloud. Queste problematiche possono essere mitigate da un service provider:

- tramite soluzioni applicative o middleware (Google, per esempio, fornisce degli strumenti per poter lavorare offline sull'email e sugli altri strumenti offerti)

- tramite la fornitura di un servizio di rete sottoposto a un Service Level Agreement (SLA), che può essere rispettato sfruttando una connettività dedicata per l'accesso oppure basandosi su una connessione Internet con uno SLA specificato.

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

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

1.6.2 Privacy e sicurezza dei dati

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

Per quanto riguarda la privacy, la normativa del settore è abbastanza lacunosa e non in linea rispetto alle evoluzioni tecnologiche, per cui il giudizio spesso può dipendere dalle singole interpretazioni. La legge vigente è comunque quella del paese dove è localizzato il datacenter e solo alcuni provider permettono poi di scegliere quello da utilizzare: questo permette di poter valutare dove ci siano maggiori garanzie e un'azienda italiana, per esempio, potrà scegliere un datacenter in Germania consapevole che i dati memorizzati in tale paese saranno protetti dalla legislazione europea. Rimane comunque il problema in alcuni settori, quello sanitario ad esempio, nel quali la legislazione vigente sconsiglia o impedisce il trasferimento di dati al di fuori dei confini italiani. In questo caso e, in genere, in tutti i settori nei quali le esigenze di privacy e sicurezza sono molto elevati, si preferisce

orientarsi verso scenari di hybrid o private cloud, che, pur avendo costi iniziali maggiori, permettono poi di godere di tutti i vantaggi offerti dal cloud computing.

1.6.3 Vendor lock-in

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

A livello di infrastruttura, infatti, viene in genere fornita una macchina virtuale che può essere facilmente migrata da un provider all'altro ed esistono anche formati standard come l'Open Virtualization Format (OVF)[DMT10]. Per quanto riguarda la gestione, non c'è tuttavia una API standard: quella più usata è quella proposta da Amazon. A livello di Platform e Software, invece, la maggior parte dei provider, pur sfruttando a basso livello i tipici protocolli di comunicazione di rete, espone all'utente soluzioni proprietarie; decidendo, per esempio, di adottare le API di Google App Engine, ci leghiamo strettamente a questa piattaforma, con il rischio di dover sottostare a cambiamenti arbitrari di prezzo e di doverci fidare delle garanzie degli specifici provider.

La soluzione è quella di andare verso standard aperti, con specifiche libere sulle componenti base e comuni a tutte le soluzioni cloud, mentre la concorrenza avviene sulla qualità dell'implementazione e su funzionalità avanzate ed innovative; una delle proposte più interessanti è quella di OpenStack [Ope], che propone delle API a livello IaaS rilasciate con licenza libera da utilizzare al posto delle API Amazon, attuale standard *de facto*. Un'altra alternativa è di utilizzare una libreria che astragga dalle particolarità delle singole piattaforme di virtualizzazione, come per esempio libvirt[3], rilasciata sotto una licenza di tipo open source.

Il secondo problema del vendor lock-in è quello dei dati: è inutile, infatti, avere delle API standard se poi non riusciamo a migrare dati esistenti da un provider all'altro. Per ovviare a questo problema, Borenstein e Blake sostengono la necessità di introdurre standard valutativi del cloud, che, similmente agli standard ISO 9000, valutino l'affidabilità di un'azienda e la sicurezza che essa garantisce[BB11].

[3] http://libvirt.org/

1.6.4 Gestione delle risorse

La gestione dinamica delle risorse rimane un problema ancora molto complesso, dal punto di vista sia del datacenter sia dell'applicazione finale. Se infatti, come si è detto poco fa, uno dei vantaggi del cloud per l'utente del servizio è quello della visione dell'hardware potenzialmente infinito, non va dimenticato come a livello fisico ci sia, in ogni caso, dell'hardware, che va scalato e suddiviso tra i vari utenti mediante tecniche di prenotazione delle risorse future e di monitoraggio delle risorse esistenti. Particolarmente difficoltosa è la gestione dello storage, che deve poter offrire una grande scalabilità e elasticità (spesso richiedendo la creazione di nuovi algoritmi e sistemi di controllo), senza però perdere di vista la protezione dei dati conservati. Lo stesso dimensionamento complessivo di un datacenter non è un problema indifferente, in quanto va garantito un buon compromesso tra numero di utenti servibili e investimenti necessari, supportando la flessibilità richiesta dal paradigma cloud.

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

Prendendo in considerazione il lato dell'utente, va sottolineato come i provider garantiscano solo dei parametri indicativi per quanto riguarda CPU, I/O e connessione; nei momenti di maggior carico e in base alle ottimizzazioni dei vari *hypervisor*, le prestazioni, a livello di una singola macchina virtuale, potrebbero ridursi e quindi è necessario tenere in considerazione anche questo fattore se le applicazioni sul cloud hanno dei requisiti per questi elementi.

1.6.5 Licenze dei software

Le licenze attuali dei software commerciali hanno molto spesso delle restrizioni sul numero di computer sui quali esso può essere in esecuzione; questo ovviamente pone dei problemi in un ambiente cloud, dove si applicano modelli di pagamento diversi e diventa difficile valutare il numero di computer o di utenti. Per questo motivo, i software sono principalmente resi disponibili con due tipi di licenze in ambiente cloud: *pay as you go*

(PAYG), in cui si paga in base al numero di ore di utilizzo oppure *bring your own licence* (BYOL), nel quale è l'utente a doversi preoccupare di disporre di licenze compatibili con l'uso in cloud. Il primo modello è, ovviamente, quello più moderno e facilita l'utente, ma al momento dipende in larga base da accordi specifici tra cloud provider e produttori di software: per fare un esempio, Microsoft SQL Server è disponibile con una licenza PAYG su Amazon EC2, mentre su IBM Smart Cloud questo modello di licensing è offerto principalmente per software di IBM stessa e per pochi altri software di produttori indipendenti. Per quanto riguarda il secondo modello, nascono diversi problemi che sono emersi con lo sviluppo della virtualizzazione, tecnologia come detto alla base del cloud: in questo caso non abbiamo infatti una visione diretta dell'hardware sottostante e il modello di pagamento più comune, quello in base al numero delle CPU (o dei socket), perde chiaramente di senso e rischia di ridurre i vantaggi offerti dalla nuova tecnologia.

Dal lato degli sviluppatori di software il nuovo paradigma pone nuove problematiche di licensing sia dal punto di vista strettamente economico che tecnico; quest'ultimo problema è stato affrontato nel corso di diversi studi, fra i quali [DP09] nel quale si propone GenLM, un sistema di management che sfrutta la crittografia asimmetrica, per consentire al provider di autorizzare e fatturare il servizio a livello di singola operazione, situazione piuttosto comune in un ambiente di grid e cloud computing. Ciò avviene calcolando l'hash dei dati in input e firmando poi tale hash, che viene quindi rimandato all'utilizzatore per poter essere mandato in esecuzione.

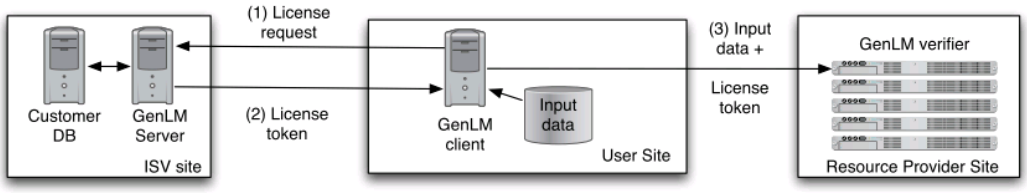

Figura 1.3: I componenti di GenLM

Capitolo 2

Soluzioni IBM per il cloud computing

IBM, tramite i propri servizi sia di outsourcing che professionali e i propri prodotti, è in grado di offrire ai propri clienti una soluzione completa di cloud computing, in grado di spaziare dalla virtualizzazione del desktop fino ad applicativi dedicati alla Business Intelligence.

2.1 Evoluzione

L'ingresso di IBM nel mondo del cloud si può far risalire al 15 novembre 2007, quando venne annunciata l'iniziativa "Blue Cloud", che si basava, a sua volta, sulla precedente soluzione di "On Demand Business", annunciata dal CIO di IBM Sam Palmisano nel 2005. "On Demand Business" si configurava come un cambiamento fondamentale nell'approccio all'architettura computazionale, che, dalle mere funzioni tecnologiche, si avviava verso delle soluzioni integrate e offribili come servizi, nelle quali non veniva più data la massima importanza alla tecnologia di base.

L'introduzione del cloud, secondo IBM, parte da una serie di motivazioni che si possono riassumere in:

tecnologiche: aumento della banda disponibile in corrispondenza di una riduzione del suo costo, proliferazione di standard aperti, maggior consapevolezza generale degli standard di sicurezza

economiche: le aziende comprendono sempre di più come i costi per le infrastrutture siano in continuo aumento, per il costante incremento di richieste di spazio e capacità

di calcolo; parallelamente si cerca di ridurre al minimo il tempo richiesto per portare le soluzioni sul mercato

culturali: gli utenti pretendono sempre più flessibilità e possibilità di accedere ai servizi in ogni momento; in generale hanno maggiori richieste verso le aziende

A partire dall'esperienza di Blue Cloud è stata introdotta la prima offerta cloud commerciale di IBM, annunciata il 16 giugno 2009 con il nome di Smart Business Cloud; essa si pone alla base dell'attuale ecosistema Smart Cloud, che presenteremo tra poco.

2.2 Cloud Computing Reference Architecture

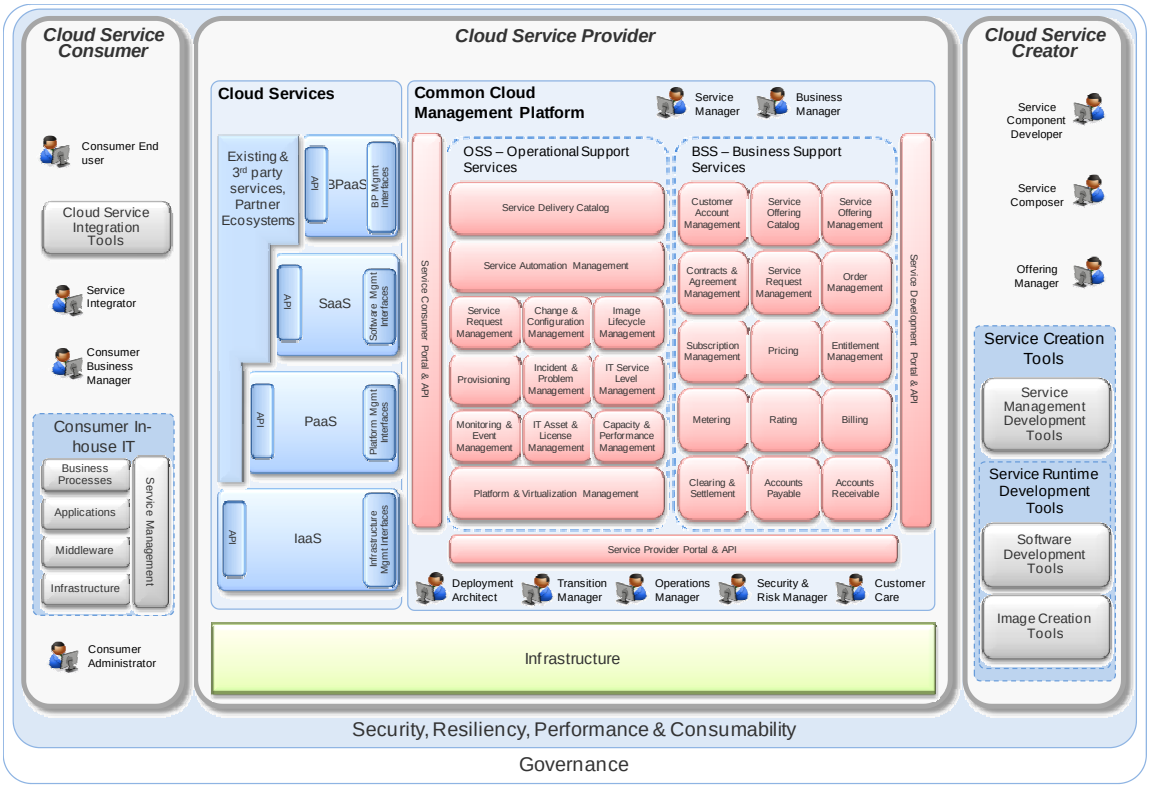

Figura 2.1: IBM Cloud Computing Reference Architecture Overview Diagram

L'offerta IBM si basa su una reference architecture[1] (CCRA) [IBM11], descritta da un documento IBM disponibile al pubblico e dalla quale si può partire per la creazione di nuove soluzioni basate su di essa.

2.2.1 Architettura logica e fisica

Nel diagramma generale della reference architecture, mostrato in figura 2.1, possiamo vedere definiti gli elementi alla base di un qualsiasi ambiente per la fornitura di servizi cloud. Sono definiti tre ruoli, discussi di seguito in dettaglio; il diagramma definisce poi i servizi cloud offerti e i servizi di amministrazione richiesti per il mantenimento dell'intera struttura. Tali servizi di management non formano un blocco monolitico, ma sono in genere implementati per mezzo di una serie di prodotti software integrati tra loro e si possono a loro volta suddividere in una parte di supporto alla parte operazionale e una parte di supporto alla parte di business. L'architettura della CCRA è utilizzabile per fornire un servizio a qualsiasi livello dello stack, dalla pura infrastruttura (IaaS) fino a uno scenario di Software-as-a-Service.

L'architettura ha un livello logico e uno fisico; il livello logico descrive le componenti con i requisiti e le funzionalità necessarie, che permetteranno la scelta dei prodotti hardware e software, che ne costituiranno il livello fisico. Naturalmente se il livello logico è unico, possono esistere numerose implementazioni fisiche, una per ogni azienda che sceglie di adottare la Reference Architecture per fornire i propri servizi cloud e su tale parte ci può essere quindi la concorrenza, in modo analogo ad altri standard aperti come J2EE. In figura 2.2 vediamo i software scelti da IBM per la propria architettura fisica.

2.2.2 Pattern

Una volta scelta l'architettura fisica, si possono ancora presentare più possibilità di applicarla. Studiando il caso specifico di IBM, possiamo evidenziare come esistano diversi pattern di implementazione, fra i quali quello utilizzato per implementare le architetture delle soluzioni SCE, che descriviamo in maggior dettaglio nelle sezioni successive. Un pattern alternativo, ma sempre sviluppato sulla CCRA, è alla base della soluzione Smart Cloud Enterprise+.

[1] IBM definisce una Reference Architecture come un documento che descrive un'architettura end-to-end in uno specifico dominio di interesse

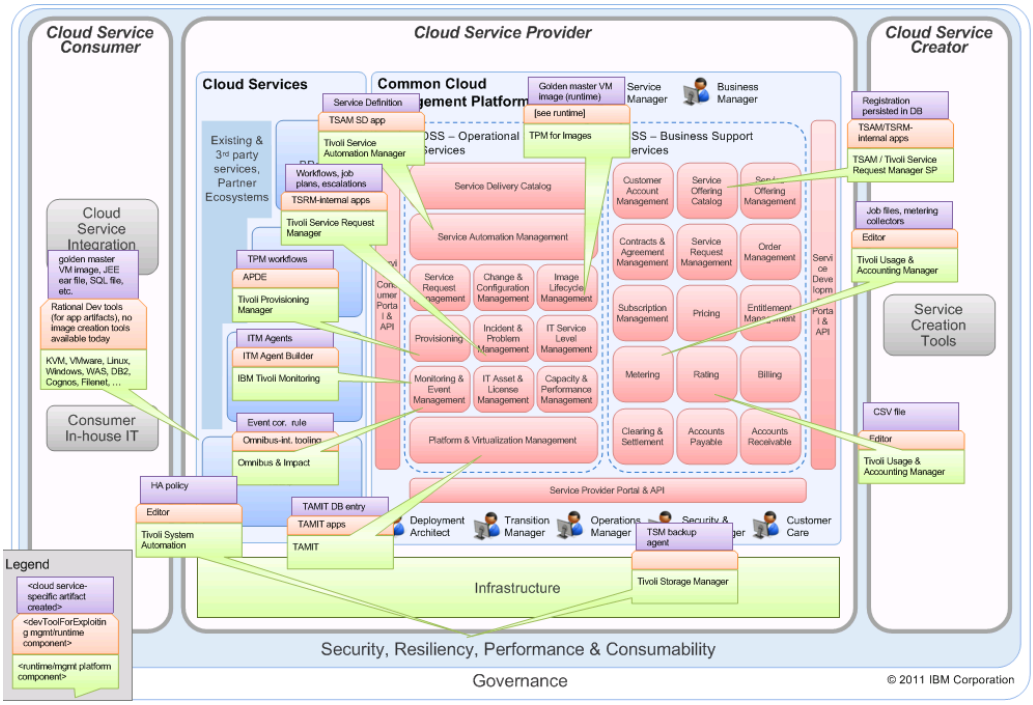

Figura 2.2: Cloud Computing Reference Architecture con il livello fisico definito da IBM

2.2.3 Ruoli

La RA prevede tre ruoli, che possono anche sovrapporsi: il cloud consumer, il cloud provider e il cloud creator; esterni a questi tre ruoli ci sono la parte di gestione della sicurezza, la resilienza e altre caratteristiche generiche del sistema completo. Può essere affrontata dal punto di vista sia di un'attività completa (vengono per esempio spiegati i dodici passaggi necessari per avviare un'infrastruttura di tipo cloud, con i prodotti corrispondenti a ognuno di essi, con un occhio di riguardo per quelli IBM), sia dei singoli aspetti (per esempio la virtualizzazione o la gestione di più utenti).

In termini pratici, la RA è una raccolta di documenti che si soffermano sulle parti fondamentali di figura 2.1; ne vediamo ora più in dettaglio alcune sezioni, suddivise in base ai ruoli.

2.2.3.1 Cloud service consumer

Questo ruolo individua i compiti che spettano all'utente del servizio offerto: in fase di progettazione essi includono lo sviluppo dell'integrazione di un'infrastruttura IT esistente

all'interno di una azienda con il mondo cloud; questa parte viene dunque maggiormente sviluppata in progetti di tipo hybrid cloud. Si richiedono in queste situazioni strumenti quali i Cloud Service Integration Tools, che invece vengono trascurati in soluzioni di tipo consumer, dove la necessità di interazioni con servizi preesistenti è meno forte. Soluzioni di questo tipo sfruttano spesso architetture ben collaudate come l'Enterprise Service Bus, che si stanno rapidamente estendendo verso l'integrazione di soluzioni cloud; per fare un esempio IBM, la soluzione offerta è Cast Iron, che si integra nativamente con WebSphere ESB, la soluzione generale di integrazione dell'azienda americana.

Terminata la fase di sviluppo, vengono anche previsti sotto questo ruolo i compiti che spettano al responsabile della gestione dell'infrastruttura creata (administrator); una volta installato il tutto, infatti, sono anche previsti nella RA degli strumenti appositi che l'amministratore del sistema può utilizzare per individuare problemi dal lato tecnico e gestire la parte economica del servizio.

2.2.3.2 Cloud service creator

In questa sezione sono definiti gli strumenti forniti a supporto di coloro che decidono di implementare per proprio conto un servizio cloud. In particolare, nella Reference Architecture, vengono specificati i Service Management Development Tools, che prevedono l'implementazione di artefatti di gestione specifici per il servizio cloud, per esempio agenti di monitoraggio. Un altro esempio è costituito dai Service Runtime Development Tools, che sono relativi allo sviluppo di artefatti relativi alla fase di runtime, per esempio per lo sviluppo di un'immagine con preinstallato un software da fornire come servizio (SaaS).

2.2.3.3 Cloud service provider

Il livello del provider è ovviamente quello più ampio e complesso, per cui viene diviso in più sezioni: la parte di infrastruttura, i servizi cloud forniti ai diversi livelli e il Common Cloud Management Platform, che viene a sua volta divisa in parte operazionale relativa al runtime (OSS), che si occupa di gestire i componenti in esecuzione sul sistema, ad esempio il ciclo di vita delle immagini e la parte di business (BSS), dedicato alla gestione commerciale.

2.2.3.3.1 Cloud services In questa parte vengono inclusi i livelli architetturali previsti e le API disponibili per interfacciarsi con il mondo esterno.

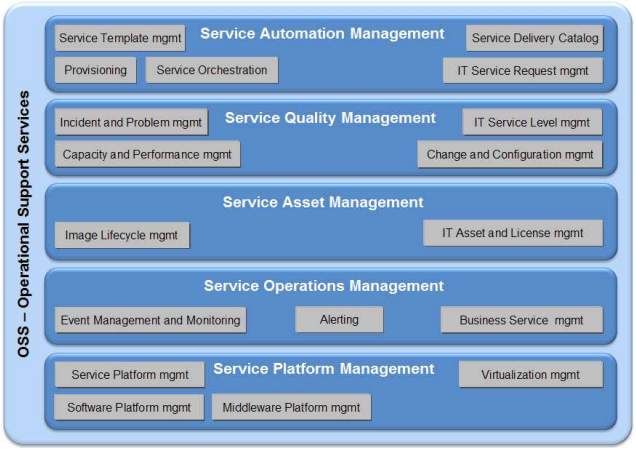

Figura 2.3: Operational Support Services

2.2.3.3.2 Operational Support Services Come detto, questa parte raccoglie tutti i servizi necessari al supporto del runtime; vediamo nella figura 2.3 una possibile suddivisione in base ai compiti pertinenti a questo livello.

Partendo dal livello più basso, troviamo la gestione dei servizi di piattaforma, fra cui la virtualizzazione (e quindi la gestione delle risorse virtualizzate) con la spinta verso soluzioni aperte quali il formato OVF; salendo, troviamo la gestione dei servizi operativi con la gestione degli eventi e delle notifiche. C'è poi la gestione degli asset, fra i quali il ciclo di vita delle immagini; la gestione della Quality of Service, che include il monitoraggio di prestazioni, problemi e raggiungimento dello SLA previsto e infine quella dell'automazione del servizio, che include parti fondamentali e complesse come quelle del provisioning delle risorse oltre che l'intero tema dell'orchestrazione dei servizi.

2.2.3.3.3 Business Support Services Per quanto riguarda i servizi di supporto al business, possono venire suddivise nelle categorie previste in figura 2.4, senza pretese di avere un ordinamento preciso. Troviamo innanzitutto la parte di gestione del cliente; c'è poi la parte di gestione delle offerte, che includono il catalogo di tutte le risorse disponibili ognuna con il loro prezzo. Un'altra categoria prevista è quella della gestione delle sottoscrizioni, che contiene, fra le altre cose, la gestione degli ordini e una parte molto importante come

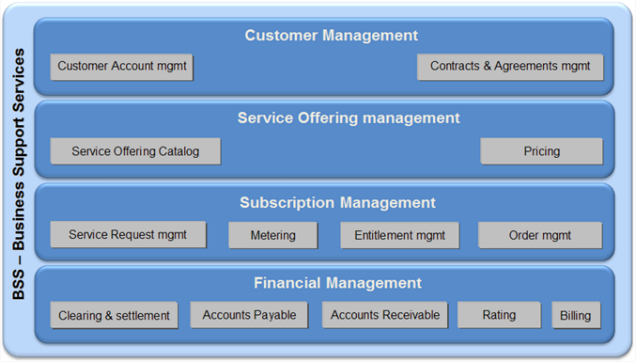

Figura 2.4: Business Support Services

la misurazione dell'utilizzo dei servizi. L'ultima parte prevista all'interno di questa suddivisione è quella più propriamente finanziaria, che include la gestione dei pagamenti e della fatturazione.

2.2.3.3.4 Infrastruttura Questa parte include tutta la parte hardware che è alla base del cloud computing, suddivisa ulteriormente in server, storage, network e facilities (per esempio la locazione fisica del datacenter e l'energia necessaria al suo funzionamento).

2.2.4 Topologia a livello Enterprise

L'architettura CCRA può essere implementata da più data center che ospitano le funzionalità di BSS e OSS, anche in maniera differenziata. Per quanto riguarda IBM, la sua offerta cloud prevede un unico centro di gestione BSS che amministra diversi datacenter secondari (chiamati Point of Delivery), sui quali troviamo i componenti OSS.

I datacenter IBM sono poi stati posizionati in luoghi in grado di offrire buoni tempi di accesso e basse latenze al maggior numero possibile di utenti; per esempio, il datacenter tedesco è posizionato nelle vicinanze di Stoccarda sulla dorsale di connessione europea in modo da offrire buon supporto a collegamenti da gran parte del continente europeo.

2.2.5 Caratteristiche comuni

Tra le caratteristiche comuni ai vari livelli, raggruppate in basso in figura 2.1, ci sono l'attenzione per la sicurezza, per la quale si suggerisce di porre particolare attenzione sul

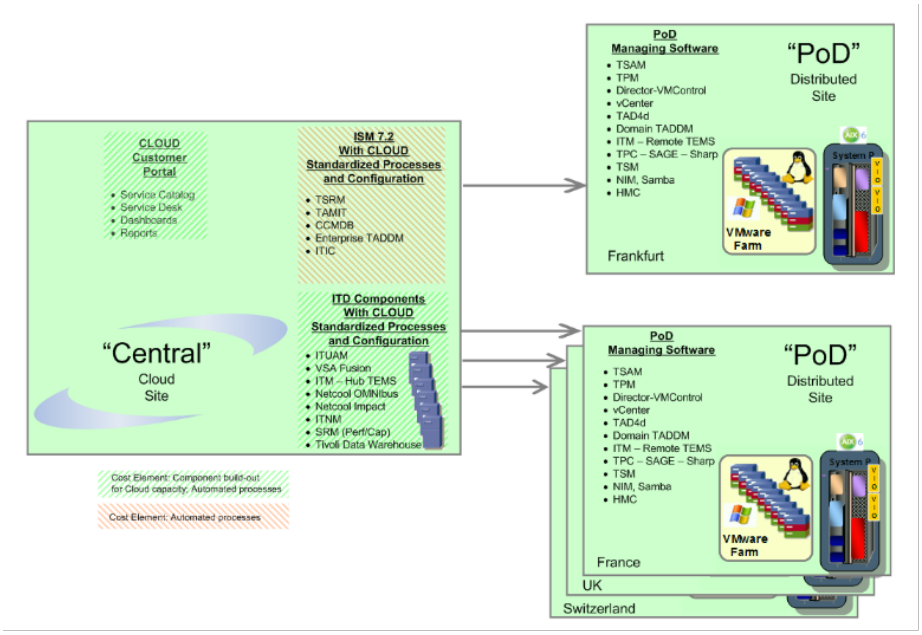

Figura 2.5: La topologia scelta da IBM

design per la multi-tenancy e sull'isolamento tra le varie risorse gestite, la resilienza, sia dei dati sia delle configurazioni, che impone di prendere in considerazione scenari di High Availability sin dalla fase di progetto. C'è poi la consumability, con cui si intende tutto ciò che riguarda l'esperienza dell'utente: facilità di configurazione, ampiezza del catalogo, adattabilità alle soluzioni esistenti e ai requisiti posti dal cliente. Anche quest'ultima parte viene posta esternamente ai tre ruoli considerati, in quanto abbraccia l'intero panorama.

Infine troviamo l'aspetto prestazionale, che spesso ha degli obiettivi contrastanti tra i tre ruoli previsti; in questa parte viene posta particolare attenzione alla fase di provisioning, che è tra quelle più complesse e che si cerca di risolvere con la scalabilità orizzontale dei componenti di automazione, il caching delle immagini virtuali e il supporto per domini di amministrazione della virtualizzazione multipli, anche con tecnologie diverse alla base (KVM, VMware, ecc.).

2.3 Smart Cloud

Sotto questo marchio IBM riunisce tre segmenti di offerta:

1. Smart Cloud Foundation, rivolto principalmente all'offerta di strumenti per la creazione rapida di cloud privati e ibridi
2. Smart Cloud Services, la piattaforma principale che presenta l'offerta IBM nell'area del public cloud
3. Smart Cloud Solutions, dedicato all'ambito del SaaS e BPaaS.

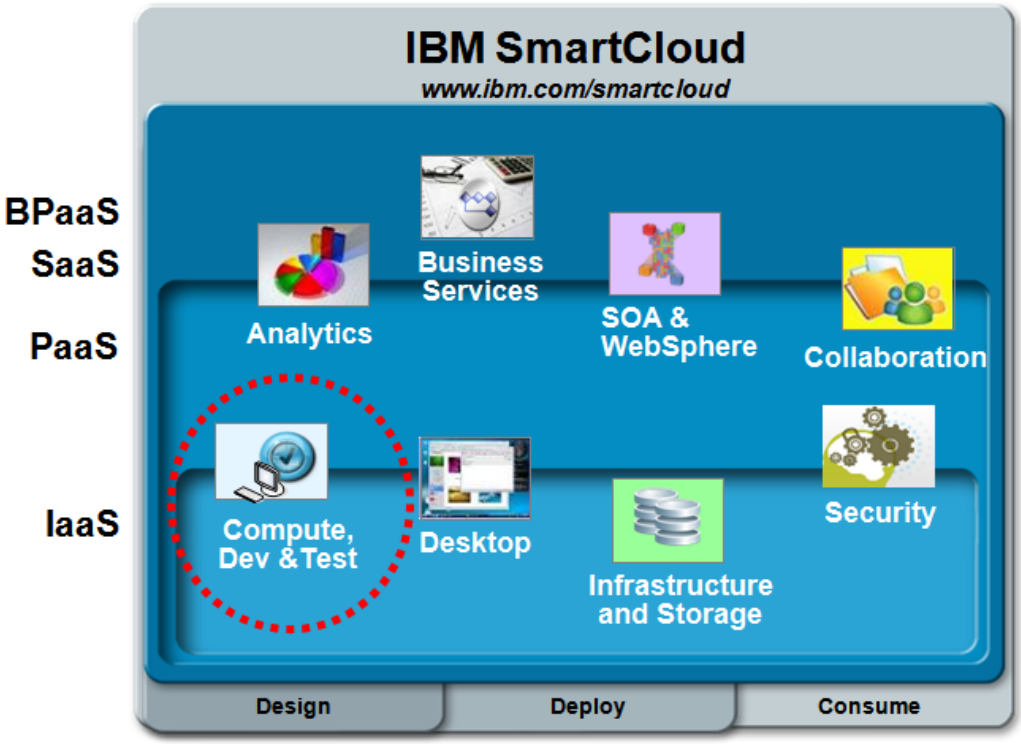

Figura 2.6: I livelli previsti dalla soluzione IBM Smart Cloud, con cerchiata la parte definita Smart Cloud Enterprise

Vediamo ora più in dettaglio la seconda di queste tre parti, che sarà anche quella sulla quale verterà la parte sperimentale di questo lavoro di tesi.

2.4 Smart Cloud Enterprise

All'interno degli Smart Cloud Services troviamo Smart Cloud Enterprise (SCE), che si posiziona nell'ambito del public cloud. È un'offerta prevalentemente di tipo IaaS, presentata

ufficialmente il 7 aprile 2011, dove gli utenti finali sono le imprese in possesso di partita IVA che sottoscrivono un contratto con IBM; non è invece possibile l'acquisto tramite carta di credito come per altri provider. Con questo servizio è possibile creare delle istanze sulla base di un catalogo predeterminato, riportato in tabella 2.1, che permette di fornire il servizio IaaS mediante:

- immagini virtuali fornite con quattro versioni di sistemi operativi (Windows Server 2003, Windows Server 2008, Red Hat Enterprise Linux, Suse Linux Enterprise Sever) a 32 e 64 bit e svariate opzioni per quanto riguarda numero di CPU virtuali, dimensione della RAM e spazio disco a disposizione

- altre componenti di infrastruttura quali storage persistente, indipendente dalla singola istanza e la possibilità di avere IP fissi oppure una Virtual Private Network.

Le immagini sono istanziabili a piacere dall'utente nei 6 datacenter previsti: tre nel Nord America, uno in Germania, uno in Giappone e infine uno a Singapore. Il servizio si estende verso il SaaS grazie alla presenza nel catalogo di immagini, che includono al loro interno software preinstallato sia di IBM sia di aziende esterne. IBM fornisce anche dei servizi di assistenza remoti o presso il cliente, per aiutarlo nell'utilizzo della soluzione e nello spostamento di applicazioni su di essa.

VM a 32 bit	Copper	Bronze	Silver	Gold
CPU virtuali a 1.25GHz	1	1	2	4
Memoria virtuale (Gigabyte)	2	2	4	4
Storage istanza (Gigabyte)	60	175	350	350

VM a 64 bit	Copper	Bronze	Silver	Gold	Platinum
CPU virtuali a 1.25GHz	2	2	4	8	16
Memoria virtuale (Gigabyte)	4	4	8	16	16
Storage istanza (Gigabyte)	60	850	1024	1024	2048

Tabella 2.1: I tagli di istanze disponibili per Smart Cloud Enterprise

2.4.1 Architettura e risorse di base

2.4.1.1 Architettura

A livello architetturale, la soluzione risulta un'applicazione della Reference Architecture e risulta così suddivisa:

- un componente BSS di gestione del servizio, monitoraggio, accounting che include l'infrastruttura centrale di supporto del Portale IBM Web Cloud e le informazioni di contatto dell'Account del Cliente; questa componente è posizionata nella sede centrale del data center del sistema di supporto al servizio, a Raleigh (Stati Uniti)
- diversi point of delivery (al momento ne sono attivi sei, ma sono in programma delle espansioni future), che contengono gli agenti BSS per la gestione dalla sede centrale, le componenti OSS e l'infrastruttura fisica.

2.4.1.2 Infrastruttura

La configurazione dell'infrastruttura fisica per la soluzione SCE è stata disegnata al fine di garantire capacità elaborativa a un costo ridotto e con una affidabilità di accesso alle macchine virtuali del 99,5%.

Dal punto di vista dell'hardware, sono stati scelti dei server IBM System X Dataplex ad alta densità. Per quanto riguarda la connettività, ogni macchina può essere collegata direttamente a Internet, avendo assegnato da 1 a 4 indirizzi IP pubblici oppure si può avere un collegamento di tipo Virtual Private Network (VPN). Tale servizio consiste in una VLAN privata presso il datacenter IBM prescelto ed un gateway VPN per accedervi. In questo modo un'istanza può essere acceduta solo attraverso il gateway da parte di utenti autorizzati; il requisito lato client sarà quello di avere un gateway IPSec sulla rete dalla quale avviene l'accesso. Dal punto di vista fisico, la connettività è garantita da due interfacce di connessione per ogni server, entrambe di tipo Gigabit Ethernet, una utilizzata per l'utilizzo dello storage e il management, l'altra per il collegamento da parte del cliente.

Lo storage può essere di tipo *ephemeral*, legato alle istanze oppure di tipo persistente, attraverso una soluzione di tipo NAS, che viene attaccata via NFS all'hypervisor; quest'ultimo, in particolare, sfrutta la tecnologia SONAS [KGP+], introdotta dall'IBM, che, fra le sue caratteristiche principali, include alta scalabilità (fino a 21 petabyte), efficienza nella gestione dei file, strumenti avanzati di amministrazione e disaster recovery con il fine di

raggiungere un'alta disponibilità. Una soluzione di questo tipo è stata scelta perché offre grossi vantaggi dal punto di vista dei costi rispetto ad avere storage separati per le singole macchine.

Per ciò che riguarda lo strato software, le macchine guest sfruttano come hypervisor la soluzione opensource KVM; viene inoltre utilizzato TSAM (Tivoli Service Automation Manager) che gestisce tutta la parte di provisioning e amministrazione delle istanze, mentre un database IBM DB2 conserva i metadati sulle immagini. Vengono poi sfruttati dei componenti della famiglia IBM WebSphere, i quali si interfacciano con il motore di reporting e analytics, garantiscono il load balancing sui diversi nodi e presentano contenuti statici e interfaccia di gestione del servizio agli amministratori del sistema.

2.4.2 Uso del servizio

Il servizio viene gestito tramite un pannello di controllo *Web based* visibile in figura 2.7 che permette la gestione di istanze, immagini e storage aggiuntivo attraverso pochi passaggi. Una volta creata una macchina virtuale, vi si può accedere in maniera diversa in base al sistema operativo: nel caso di istanze Windows, l'accesso avviene nell'interfaccia grafica attraverso Remote Desktop Connection fornito con i sistemi operativi Microsoft; se le istanze, invece, sono di tipo Linux, l'accesso può essere fatto da riga di comando attraverso una sessione SSH oppure per via grafica tramite una connessione che sfrutta il sistema VNC. Nel caso del protocollo SSH, la sicurezza viene gestita tramite crittografia asimmetrica per cui dal pannello di controllo Web si crea una coppia di chiavi, una pubblica (che viene inserita nell'istanza) e una privata (da usare al momento della connessione); negli altri casi si usano, invece, una coppia nome utente e password, che vanno specificati al momento della creazione dell'istanza.

2.4.3 API

Per l'offerta Smart Cloud Enterprise, IBM mette a disposizione delle Application Programming Interface, che replicano completamente tutto ciò che si può fare dal pannello di controllo citato poco sopra e permettono anche alcune opzioni aggiuntive in fase di configurazione (per esempio c'è la possibilità di creare delle istanze fisicamente separate in uno stesso datacenter). L'API di base è quella RESTful, sulla cui base ne vengono messe

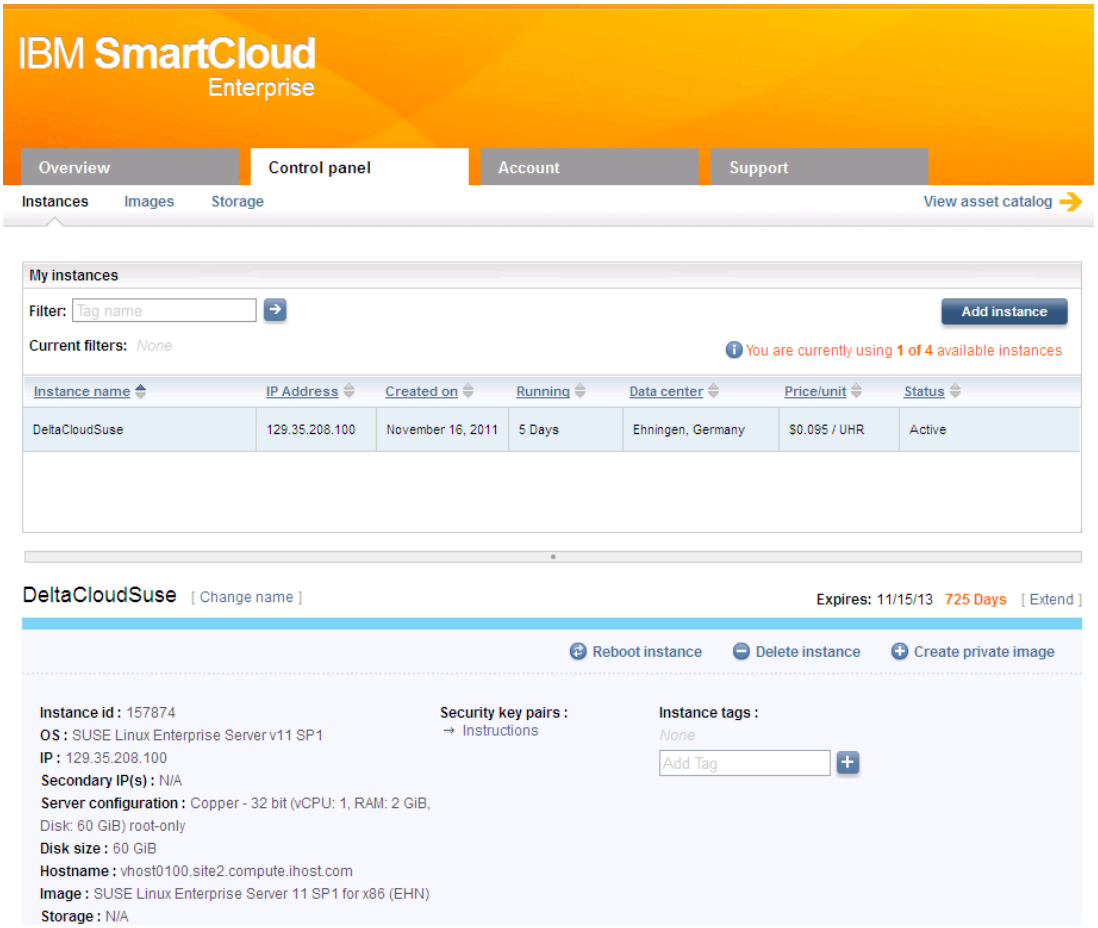

Figura 2.7: Il pannello di controllo web di Smart Cloud Enterprise

a disposizione altre due: Java e riga di comando, in grado di funzionare con sistemi operativi sia Microsoft Windows che Linux. L'API RESTful è costituita da chiamate HTTP GET (per ricevere informazioni) e POST a un application server posto sul cloud IBM; ciò apre anche la possibilità di avere dei pannelli di amministrazione personalizzati, che offrano solo determinate funzionalità, per mezzo della creazione di un proxy attraverso il quale passano le richieste. Possiamo suddividere le funzionalità delle API nelle seguenti categorie:

- gestione (creazione, modifica e eliminazione) delle istanze e informazioni su di loro
- gestione (creazione, modifica e eliminazione) di immagini e informazioni su di loro
- gestione e informazioni sullo storage

- gestione di indirizzi riservati e VLAN
- informazioni su prezzi e locazioni del servizio
- informazioni sull'utilizzo di risorse e la fatturazione degli utenti gestiti.

Dal punto di vista della compatibilità, bisogna notare come IBM abbia scelto una via proprietaria rispetto allo standard de facto che sono le API di Amazon EC2/S3; anche se le funzioni di base sono le stesse, questa compatibilità potrebbe rendere difficile la migrazione sia verso piattaforme compatibili con Amazon che verso altre soluzioni, a meno di non utilizzare sin dall'inizio soluzioni che astraggano dalle differenze come DeltaCloud[2]. IBM dichiara comunque che, nelle prossime versioni, verranno implementate soluzioni standard, che per ora non sono ancora emerse con chiarezza. Per fare un confronto, altri provider IaaS fanno scelte differenti: Eucalyptus, una delle soluzioni più utilizzate nell'ambito private cloud, espone API pienamente compatibili con quelle di Amazon AWS[3], mentre la soluzione opensource OpenStack espone delle API proprietarie le cui specifiche sono aperte, ma supporta anche le API Amazon per motivi di compatibilità.

2.4.3.1 Esempi

Vediamo un rapido esempio delle API che permettono di recuperare informazioni sulle istanze attive al momento, che ha la seguente signature:

```
GET <base_URL>/instances
```

Essa fornisce nella versione Java una serie di informazioni in output di questo tipo:

```
Found 2 instance(s).
Name: DeltaCloudSuse; Type: COP32.1/2048/60; Location:
    Ehningen, Germany; IP Address: 129.35.208.100; Key name:
    ChiaveProva; Disk size: 60
Name: provalib; Type: BRZ32.1/2048/60*175; Location:
    Ehningen, Germany; IP Address: 129.35.214.169; Key name:
    ChiaveProva; Disk size: 235
```

[2] http://incubator.apache.org/deltacloud
[3] http://www.eucalyptus.com/resources/AmazonAWS

Un'altra API rilevante è quella per la creazione di nuove istanze, che ha questa signature:

POST <base_URL>/instances

A differenza che nel caso precedente, è necessario fornire una serie di parametri:

- NAME: nome dell'istanza da creare
- IMAGEID: ID dell'immagine dalla quale creare l'istanza
- INSTANCETYPE: taglio dell'istanza
- LOCATION: codice del datacenter prescelto
- IP: (opzionale) indirizzo IP riservato da usare come primario
- VOLUMEID: (opzionale) ID dell'unità di storage persistente da associare all'istanza
- VLANID: (opzionale) ID della VLAN da associare all'istanza
- SECONDARYIP: (opzionale) indirizzo IP riservato da usare come secondario
- ISMINIEPHEMERAL: (opzionale) se vero, parte con uno storage integrato minimo, rendendo la creazione più rapida
- OSS.STORAGE.ID.0.MNT: (opzionale, Linux) permette di specificare un point personalizzato per lo storage
- ANTICOLLOCATIONINSTANCE: (opzionale) ID di un'istanza da cui quella in creazione deve essere fisicamente separata
- PUBLICKEY: (opzionale) nome della chiave pubblica che permette l'accesso all'istanza. Necessario per istanze Linux
- CONFIGURATION DATA: (opzionale) ulteriori parametri specifici per le singole immagini. Nel caso di istanze Windows, è necessario specificare UserName e Password per accedere.

2.4.4 Funzionalità avanzate

Una delle direzioni di sviluppo più recenti della piattaforma SCE è quella della composizione di immagini: sono state infatti messi a disposizione degli strumenti in grado di partire da un'immagine disponibile a catalogo e aggiungere in poche fasi software o middleware fino ad arrivare a una nuova immagine, che è poi possibile inserire nel catalogo e mettere a disposizione degli utenti del servizio. Questo è molto interessante in un ottica SaaS, riducendo il tempo necessario per il deployment e permettendo una crescita del catalogo delle immagini, la cui ricchezza è uno dei punti di forza di Amazon Web Service; nel suo catalogo sono disponibili più di 1000 Amazon Machine Image (AMI).

Tutte le immagini a catalogo, siano esse fornite da IBM o create dagli utenti, sono conservate in SCE sotto forma di *asset* e gestite per mezzo del software Rational Asset Manager (RAM) di IBM; grazie a ciò, è possibile disporre di alcune funzioni accessorie ma interessanti, tra le quali si segnalano delle API in diversi linguaggi che permettono di effettuare una ricerca all'interno del catalogo in base a diversi parametri.

Un'altra caratteristica un po' nascosta all'interno del sito di supporto DeveloperWorks[4] è quella che ci consente di creare un'immagine con dei parametri definibili dal creatore dell'immagine stessa. Sempre sfruttando le potenzialità messe a disposizione da RAM, è infatti possibile definire una serie di script shell che vengono richiamati al momento del primo avvio dell'istanza e che possono andare a modificare a piacimento le impostazioni del sistema, inserendo i parametri scelti dall'utente al momento della creazione dell'istanza.

2.4.4.1 Object storage

A partire dall'inizio del 2012, IBM offre ai clienti del servizio SCE, accanto a quelli esistenti, un terzo metodo di storage, denominato Object Storage, che si pone in concorrenza con soluzioni quali Simple Storage Service (S3) di Amazon. Tale funzionalità viene presentata da IBM come uno spazio di storage potenzialmente infinito, caratterizzato da numerosi metodi di accesso e garanzie sulla sicurezza dei dati memorizzati.

L'object storage di IBM sfrutta la tecnologia sviluppata dalla società Nirvanix, i cui datacenter vengono utilizzati nella fase iniziale, e si caratterizza per essere basato come singola unità sui file, a ognuno dei quali viene assegnato un identificativo univoco e se necessario dei metadati aggiuntivi. I trasferimenti possono quindi essere effettuati in maniera

[4] http://www.ibm.com/developerworks/cloud/library/cl-parameterizecloudimages/index.html

molto granulare e anche i pagamenti avvengono in base alla quantità di spazio realmente utilizzato.

Gli usi suggeriti nella prima fase per questo tipo di storage sono il backup, eventualmente anche replicato automaticamente su altri datacenter per il disaster recovery, la distribuzione globale di dati, senza necessità di doversi preoccupare dei singoli dettagli infrastrutturali e infine esigenze generali di archiviazione di dati.

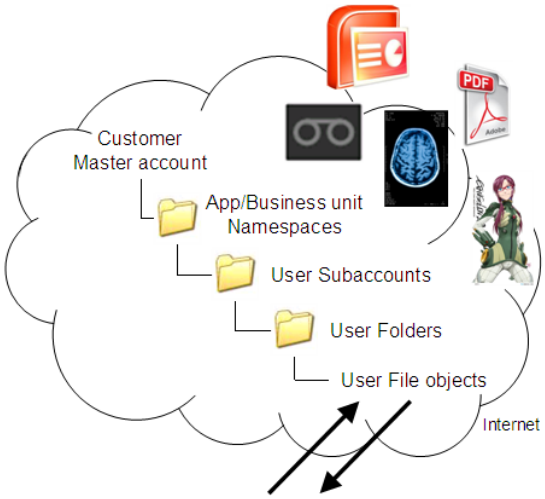

Figura 2.8: Esempio di Object Storage

I metodi di accesso previsti al momento sono delle API di tipo RESTful, dotate di *binding* per tutti i più comuni linguaggi di programmazione; vengono poi proposte delle virtual appliance, disponibili sia come soluzioni hardware sia esclusivamente software e già disponibili nel catalogo delle immagini di SCE, le quali permettono l'accesso all'object storage come se si avesse a che fare con un file system locale.

2.4.5 Limiti

La soluzione Smart Cloud Enterprise ha al momento alcune limitazioni rispetto agli altri player sul mercato IaaS, dovuti probabilmente anche alla minore maturità (la soluzione è disponibile al pubblico solamente dal 2010); tuttavia, gli sviluppi sono costanti e quindi questi limiti vengono superati o addirittura si trasformano in punti di forza delle soluzioni.

Tra i limiti principali, soprattutto in relazione al tema della nostra tesi, la totale assenza di controllo sull'Hypervisor; anche le capacità di importazione di immagini esistenti è stata introdotta nella release 2.0, ma con alcuni limiti piuttosto severi, fra i quali l'impossibilità di applicarlo per sistemi operativi Linux. Anche la procedura di copia delle immagini tra un datacenter è ancora in via di sviluppo e al momento utilizzabile solo per pochi casi.

Infine, dal punto di vista prettamente enterprise, è importante sottolineare l'assenza di Service Level Agreement molto alti, che impedisce di migrare verso SCE workload che

richiedono una grande affidabilità; al momento infatti viene garantito uno SLA del 99,5% per l'intero servizio, mentre non viene fornita alcuna garanzia sull'accesso alle istanze.

2.5 Smart Cloud Enterprise+

Smart Cloud Enterprise+ realizza una soluzione di tipo Private Cloud che viene data in outsourcing a IBM. Essa permette di coprire la fascia dei clienti che desiderano un'offerta equivalente all'outsourcing, affiancando anche i vantaggi permessi dal paradigma cloud; la sua realizzazione si avvale delle conoscenze e degli strumenti di gestione e migrazione accumulati da IBM nell'ambito dell'outsourcing. Questa duplice esigenza di avere una soluzione che risponde a SLA di affidabilità, robustezza, implementando anche un servizio Cloud permette di fornire un servizio adeguato anche ad aziende che debbano gestire dati sensibili, quali banche o aziende pubbliche.

Rispetto alla soluzione SCE viene previsto un aumento del livello di flessibilità a livello di piattaforma, con la possibilità di avere sistemi aggiuntivi di monitoraggio e funzionalità ulteriori per la sicurezza, come ad esempio il backup fornito da IBM e integrato nella soluzione; rimangono tuttavia i vincoli sulla standardizzazione dei workload e dello strato di management che sono alla base di un progetto cloud di successo. Vengono inoltre previsti nuovi *point of delivery* oltre ai sei già visti per lo Smart Cloud Enterprise e viene prevista l'opzione di avere uno SLA garantito di 99,9%, che raggiunge il 99,99% per quanto riguarda lo storage; per ogni singola istanza è possibile scegliere il livello di SLA desiderato e le garanzie sulla replicazione, così da poter prevedere utilizzi anche per fini di produzione. Dal punto di vista economico, la fatturazione avviene su base mensile piuttosto che su base oraria, come avviene in Smart Cloud Enterprise e nella maggior parte delle soluzioni di cloud pubblico.

Aumenta poi il numero di sistemi offerti: oltre a piattaforme di tipo Intel, si hanno anche piattaforme di tipo Power 7 con sistema operativo AIX, molto diffuso in ambito aziendale come backend. Il virtualizzatore alla base di Smart Cloud Enterprise+ è VMware per sistemi di tipo x86, mentre per gli IBM System p viene utilizzata la virtualizzazione PowerVM; sono previsti, infine, strumenti per l'importazione di immagini virtuali esistenti, così da facilitare il processo di migrazione.

Dal punto di vista architetturale, anche SCE+ sfrutta una diversa architettura fisica sulla base della Reference Architecture esposta in precedenza. In particolare, l'architettura

prevede una strato di managing, utilizzato per fornire dei servizi ai clienti e che consiste a sua volta in una parte centralizzata ed una distribuita nei vari point of delivery. A esso si affianca lo strato managed, che è un insieme di nodi dedicati o condivisi, che possono essere ospitati presso il datacenter IBM o direttamente presso il cliente.

Purtroppo questa soluzione sarà disponibile solamente nel secondo quadrimestre del 2012, per cui non è stato possibile fare delle prove sul campo, né sfruttarne le nuove funzionalità previste.

Capitolo 3

Valutazione dell'opportunità di migrazione

Dopo aver stabilito caratteristiche, vantaggi e limiti del cloud computing, poniamo la nostra attenzione su come si possa arrivare all'utilizzo di queste idee in modo efficiente nel mondo delle applicazioni industriali. Come si è visto, alla base del paradigma cloud, vi è una serie di concetti già esistenti e sviluppatisi in maniera separata, che sono stati integrati inoltre con nuove evoluzioni tecnologiche, quali la banda larga. Per le aziende, vista la maturità raggiunta dal settore dell'Information Technology, l'interesse maggiore è orientato a una sua introduzione per evoluzione di soluzioni preesistenti, così da permettere anche ad esse di sfruttare i vantaggi offerti (pensiamo al self-service delle risorse o alla maggior flessibilità e scalabilità), consentendo nel contempo una possibile riduzione dei costi infrastrutturali e di gestione.

Il cloud richiede di base un'idea di standardizzazione e uniformità, anche su centri di elaborazione remoti, per cui non è una soluzione che offra benefici per qualsiasi applicazione. Quindi, prima di procedere con un progetto di migrazione che coinvolga elementi già esistenti, le aziende chiedono agli esperti un'analisi che permetta il confronto con le offerte disponibili sul mercato e stabilisca, a seconda dei diversi punti di vista affrontati, cosa possa essere migrato e cosa no. Tale analisi prevede il confronto tra i workload dell'intero sistema informativo di origine e i diversi modelli di delivery cloud. Poiché i workload possono essere in numero piuttosto elevato, comprendendo nei casi più complessi anche migliaia di server, nasce l'esigenza di disporre di strumenti automatizzati, destinati a facilitare il consulente in questo compito, che sarà il tema di maggior interesse nel seguito

della trattazione.

Andremo ora a vedere lo stato dell'arte per quanto riguarda il tema della migrazione verso ambienti cloud, prima in breve dal punto di vista consumer, poi più in dettaglio l'aspetto enterprise, che sarà l'argomento centrale della tesi.

3.1 Migrazione in ambiente consumer

Dal punto di vista consumer, la migrazione è sentita soprattutto a livello di Software-as-a-Service e Platform-as-a-Service, visto che la virtualizzazione, ingrediente di base del cloud computing, non ha ancora avuto molta diffusione a livello dei singoli utenti. La maggior parte degli utilizzi è quindi relativo a nuove applicazioni, godendo della possibilità di avere dei server al momento del bisogno oppure sfruttando nuove tecniche come MapReduce/-Hadoop che, grazie alle tecniche del calcolo parallelo, riducono i tempi di elaborazione rispetto all'esecuzione su una macchina singola. Per quanto riguarda la migrazione di un software da uno scenario desktop ad uno in cloud, esistono diversi studi che si soffermano sulle difficoltà riscontrate e le modifiche necessarie sia a livello concettuale sia pratico, che di seguito andiamo ad analizzare.

3.1.1 Stato dell'arte

Per quanto riguarda la migrazione di software, non esistono soluzioni universali. Le variabili in gioco sono moltissime, partendo dallo scopo del programma fino al linguaggio utilizzato, per cui è difficile fare delle generalizzazioni. Uno studio generale sulla fattibilità della migrazione si trova in [TKLF11], dove gli autori presentano un'analisi che coinvolge tutti gli aspetti relativi alla migrazione di un applicativo *on-premise* verso il cloud, con una particolare attenzione sui fattori di costo di tale migrazione. Il test case è stato quello di un software n-tier basato sulla tecnologia .NET e gli autori hanno individuato cinque fasi necessarie: addestramento, installazione e configurazione dell'ambiente di destinazione, modifiche al codice, migrazione vera e propria e infine testing. I singoli costi di queste fasi dipenderanno poi dalle capacità del gruppo di lavoro e dalla complessità dell'applicazione in oggetto.

Un'esperienza simile è quella proposta in [BC11], dove si analizza, da un punto di vista prettamente tecnico, la migrazione verso un cloud pubblico di un'applicazione opensource.

Gli autori sottolineano innanzitutto l'importanza di suddividere il software per componenti, così da poter anche effettuare il deployment su provider diversi ed essere indipendenti anche dai singoli database. Viene anche specificato come, in ambienti di tipo PaaS, la migrazione sia più difficile e potrebbe richiedere molte modifiche perché tecnologie e librerie a disposizione sono limitate; in generale, più il sistema è complesso e dipende da molte librerie, più la migrazione diventa lunga. Gli autori concludono individuando vari fattori che influenzano la migrazione, fra i quali si segnalano la sicurezza delle comunicazioni e la difficoltà di accedere alle librerie sottostanti in un ambiente di tipo PaaS, che potrebbe rendere difficile la ricerca di bug.

In [JMW+11] si analizza invece la parte successiva alla migrazione, quella dell'ottimizzazione; nello specifico, viene studiata un'applicazione n-tier che viene migrata su due diversi provider IaaS e le prestazioni vengono confrontate con un tradizionale ambiente cluster. Il risultato interessante dello studio è che le ottimizzazioni richieste variano molto da provider a provider, in quanto le prestazioni computazionali e di rete sono molto diverse e possono anche avere una notevole variabilità nel tempo, per cui, al momento della migrazione, bisognerà porre attenzione ed effettuare dei refactoring per raggiungere gli SLA richiesti. Un interessante studio dedicato completamente alla variabilità di prestazioni di un ambiente cloud si può trovare in [SDQR10].

In [ZL] viene affrontato uno scenario n-tier simile ai precedenti, che viene migrato da un datacenter locale a Amazon EC2, attraverso la reinstallazione di tutti i componenti necessari (application server, database, ecc.) sul cloud. Gli autori analizzano gli errori riscontrati in questa fase dopo averla fatta ripetere da operatori diversi, li categorizzano e ne misurano la frequenza, per poi proporre un framework per la configurazione semi-automatica. Tale framework consiste in due parti: una di controllo periodico delle politiche di configurazione e una che offre un sistema automatico di installazione basato su modelli e quindi estendibile.

3.2 Migrazione in ambiente enterprise

A livello enterprise c'è un'interesse molto più vasto per l'argomento della migrazione verso un ambiente cloud, che, nel caso di alcuni tipi di workload, viene visto come naturale evoluzione dell'outsourcing. I datacenter dei singoli clienti assorbono infatti l'80% dei costi IT per le attività di manutenzione, riducendo enormente le risorse economiche ed umane

che possono essere investite in ricerca e sviluppo. Per questo motivo molti CIO stanno vedendo il cloud come un modello avanzato di outsourcing che possa ospitare una parte dei loro workload a dei costi più bassi di quelli raggiungibili nei loro datacenter, liberando inoltre i loro dipendenti dalle incombenze relative alla manutenzione e al supporto.

3.2.1 Stato dell'arte

Per quanto riguarda l'ambito enterprise, esso include sicuramente i case study che abbiamo visto in precedenza per l'ambito consumer. Tuttavia, in questo ambito l'interesse sull'aspetto tecnologico della questione è solo uno dei fattori presi in considerazione, mentre ne acquistano importanza altri, fra i quali uno dei più studiati è quello dell'analisi dei costi, affiancato spesso dai cambiamenti organizzativi portati da una migrazione nel cloud. Nonostante l'interesse per l'argomento sia molto alto, le analisi scientifiche e le sperimentazioni su questo tema non sono così numerose e la maggior parte delle soluzioni, in particolare gli strumenti dedicati alla migrazione, sono interni alle aziende e non ne vengono divulgate le caratteristiche tecniche e i dettagli di implementazione. Nel seguito presenteremo qualche esempio significativo riscontrato in letteratura e, successivamente, andremo a cercare altri spunti da tematiche laterali, ma affini all'argomento, come quella della server consolidation.

3.2.1.1 Studi generali dei problemi da affrontare

In [CS11] si analizza il fattore costo, che è uno dei primi tra quelli presi in considerazione in ambito aziendale, anche se, in questo caso specifico, il punto di vista è puramente tecnologico. Esso viene suddiviso in macrocategorie: hardware dei server, energia per l'intera infrastruttura di supporto, servizio (compreso il personale), hardware di rete e infine spazio fisico del datacenter. Gli autori suggeriscono che, in caso di single-tenancy con un unico utilizzo del cloud, è necessario un alto utilizzo di CPU per rendere il servizio remunerativo. Questo costo cala in uno scenario di multi-tenancy; inoltre, il networking ha un peso molto forte, in quanto la fornitura di un servizio garantito ha un alto costo, anche prendendo in considerazione l'economia di scala.

Secondo [SSR+10] il cliente di tipo enterprise si distingue per avere requisiti aggiuntivi per quanto riguarda le prestazioni, la disponibilità e la scalabilità e quindi le applicazioni di questa classe sono in genere molto più complesse. Per capire la possibilità di migrarle

sul cloud, gli autori si soffermano su tre punti: come effettuare il deployment di servizi distribuiti a larga scala nel cloud, come fornire servizi ad alta disponibilità e che cosa si può fare nel caso si verifichino problemi. Per il primo punto, vengono sottolineate due problematiche: le dipendenze tra i vari componenti, che vanno prima trovate, operazione spesso difficile in assenza di documentazione, e poi replicate nel cloud. C'è poi il desiderio di ridurre il downtime e avere quindi la possibilità di *live migration*, cioè una migrazione delle macchine virtuali tra locazioni fisiche differenti mentre esse sono in esecuzione, senza quindi necessità di spegnerle o comunque di interromperne il funzionamento, che può risultare utile anche per redistribuire le macchine virtuali nel caso di carico eccessivo su una singola macchina fisica. C'è il problema, infine, di come gestire in maniera, il più possibile automatica, eventuali evoluzioni del servizio. Per quanto riguarda l'high availability, la migrazione in un ambiente cloud in grado di garantire un SLA sul servizio riduce i potenziali problemi a errori di configurazione o del software; altre sfide a questo riguardo sono la ridondanza su ambienti cloud di provider differenti e strumenti per sfruttare lo scaling automatico offerto dal cloud. L'ultimo punto è quello della risoluzione di problemi: in questo caso, un elemento di notevole importanza è capire velocemente se il problema è dalla parte del cliente o sull'infrastruttura; secondo questo lavoro, inoltre, sarebbe utile avere a disposizione informazioni di stato dei vari servizi in tempo reale, così da avere una visione anche parziale dell'infrastruttura sottostante.

Gli autori di [KHSS10] si concentrano soprattutto su tre campi di ricerca: i cambiamenti dal punto di vista dell'organizzazione, le implicazioni sull'azienda di una fatturazione a consumo e infine i problemi legati a sicurezza e privacy dei dati. Per quanto riguarda il primo punto, una delle grandi novità del cloud è l'aspetto self-service, che riduce la necessità di passare per i livelli superiori in caso di necessità di risorse e quindi genera problemi di possibile perdita di controllo; questo vale anche per quanto riguarda il supporto, su cui non si può più controllare la priorità, come invece nelle infrastrutture interne. Sui costi, gli autori si soffermano sull'incertezza intrinseca del modello a consumo, che non garantisce dei prezzi fissi e su alcuni problemi sui metodi di fatturazione. La sicurezza, infine, è uno dei temi che stanno più a cuore nell'ambito enterprise ed è strettamente legato al tema della privacy e della gestione dei dati sensibili. Per risolvere tutti questi problemi, viene affermata l'importanza dello sviluppo di framework generali che prendano in considerazione i diversi fattori analizzati.

3.2.1.2 Discovery

La fase di discovery di un ambiente esistente è la prima necessaria per una migrazione e può essere affrontata da due punti di vista: nel primo viene mantenuto un database dell'ambiente enterprise, in genere chiamato Configuration Management Database (CMDB), che fa parte delle *best practice* suggerite dall'Information Technology Infrastracture Library (ITIL) per la fornitura di servizi IT di qualità; esso dispone di una visione aggiornata di insieme, con le varie connessioni tra i sistemi e le loro principali caratteristiche. Tale database può essere interrogato per ottenere informazioni aggiornate ed è possibile estendere le informazioni disponibili attraverso componenti opzionali che effettuano una scansione di middleware e componenti aggiuntivi. Spesso l'adozione di uno strumento di CMDB o l'aggiornamento delle informazioni già presente è uno dei primi passi adottati dalle aziende in vista di una migrazione in ambiente cloud.

Tutte le aziende più grandi hanno strumenti di questo tipo e uno degli esempi è quello di IBM Tivoli Change and Configuration Management Database (CCMDB)[MSB+07]. Questo strumento implementa le quattro caratteristiche principali richieste dall'ITIL per un CMDB:

1. RICH MODEL DATA. Il modello dei dati di Tivoli CCMDB supporta tutte le strutture di base riscontrabili in un sistema IT, chiamate Configuration Item (CI) e diversi tipi di relazioni e attributi; offre poi un linguaggio simil-SQL per le query e strumenti per la composizione di CI

2. AUTOMATIC DATA DISCOVERY AND CHANGE TRACKING. Sfruttando un sistema di sensori, CCMDB permette l'esplorazione di reti anche complesse in modo poco invasivo e, per mezzo di sensori opzionali, si può estendere l'esplorazione ad applicazioni e middleware specifici, mentre delle euristiche permettono di ricostruire le relazioni implicite dell'ambiente, così da crearne un grafo completo. Nel caso frequente di provenienza dei dati da diverse fonti, sono possibili federazioni di dati e operazioni di riconciliazione

3. VISUALIZATION OF APPLICATION DEPENDENCY. Sono disponibili appositi strumenti che creano, sulla base dei dati esplorati, dei grafi completi di tutte le dipendenze dai componenti, sia a livello dei singoli middleware, sia a livello più eleva-

to di servizi forniti, così da poter farsi immediatamente un'idea di quali elementi potrebbero risentire di un cambiamento della configurazione

4. MULTICUSTOMER SUPPORT. È possibile avere scenari sia multicustomer che multitenant sulla base di un unico CCMDB, con permessi di accesso diversi ai dati in base all'utente che vi accede

Inoltre, CCMDB offre anche supporto per INTEGRATED CHANGE AND CONFIGURATION PROCESSES in quanto la gestione dei cambiamenti di un ambiente con l'obiettivo di minimizzare gli effetti avversi, pur non appartenendo strettamente alla definizione di CMDB, vi è è strettamente legata. In particolare, nella visione del tool IBM, i dati provenienti dal discovery devono essere convalidati e autorizzati e così ogni cambiamento effettuato, in modo da mantenere uno storico delle variazioni e una gestione delle responsabilità.

Un ulteriore passo di discovery è quello di modellare con precisione le caratteristiche di applicazioni e middleware installati sul sistema e in particolare quello delle relazioni tra di esse a livello di data layer. Una soluzione che affronta questo problema è Galapagos [MDJV08], un framework che si propone di partire dalle informazioni estratte da strumenti simili al CCMDB; da queste informazioni vengono ottenuti dei Data-location template (DLT), i quali contengono informazioni riguardanti il sistema in esame. La fase successiva, come visibile in figura 3.1, è quella dell'esplorazione, tramite sensori, delle *data dependency* tra i modelli DLT, le quali vengono poi analizzate per mezzo di un algoritmo al fine di popolare un *repository* contenente le relazioni tra applicazioni e dati, che poi può venire visualizzato attraverso un apposito strumento. Questa soluzione è molto flessibile e permette di ridurre eventuali errori e la complessità nella raccolta dei dati di altri approcci, ma la creazione dei DLT è comunque abbastanza impegnativa e richiede una buona conoscenza dell'applicativo da analizzare.

3.2.1.3 Selezione delle immagini di destinazione sul cloud

Tutte le principali offerte disponibili a livello di IaaS permettono all'utente di configurare l'istanza da creare in base a due criteri:

- la dimensione o taglio, definito in genere in base al numero di CPU virtuali e alla quantità di RAM e spazio su disco offerti

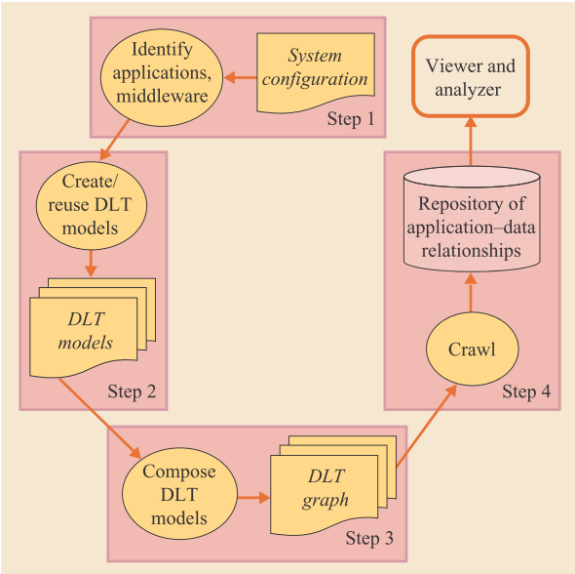

Figura 3.1: Le fasi previste da Galapagos

- il tipo di stack software da cui partire, che conterrà solitamente il solo sistema operativo per arrivare fino a livelli più alti dove middleware specifici sono già preinstallati.

Per i nostri scopi, la selezione delle immagini di destinazione ci interessa soprattutto se consideriamo la migrazione dal punto di vista della reinstallazione; in tale caso, infatti, si cerca nel catalogo l'immagine che abbia già al suo interno il software desiderato, in modo da ridurre gli sforzi dovuti alla loro installazione manuale. Per fare un esempio, nel caso della migrazione di alcuni server che ospitano un Content Management System (CMS) e che si appoggiano a un database specifico, sarà opportuno verificare, all'interno del catalogo del cloud provider, se esistano delle immagini con tali middleware già preconfigurate; in caso positivo, tali immagini potranno essere istanziate e saremo sicuri che i loro parametri di configurazione saranno ottimizzati per il cloud, per cui sarà necessario occuparsi solamente della migrazioni di dati e configurazioni specifiche.

A questo riguardo, gli autori di [FSW+10] propongono uno studio che parte dalla constatazione che, al momento, il catalogo di immagini preconfigurate più utilizzato, quello di Amazon, non contiene dei campi predefiniti che descrivano i metadati, per cui l'utente è costretto a scorrere i campi descrittivi liberi. In questo lavoro viene quindi presentato un algoritmo che, dopo aver raccolto (possibilmente tramite strumenti automatici) le informazioni sulle immagini presenti nel catalogo e i requisiti dell'utente, ne effettua un matching

tramite una funzione che prende in considerazione i costi di installazione e di aggiornamento, oltre che ai vincoli posti dall'utente e fornisce come risultato la corrispondenza migliore.

Va notato a questo riguardo come il catalogo IBM Smart Cloud, grazie al fatto di utilizzare come backend il Rational Asset Manager, disponga di API utilizzabili per la ricerca all'interno del catalogo di immagini disponibili; tali API supportano genericamente un approccio RESTful oppure il linguaggio Java[1].

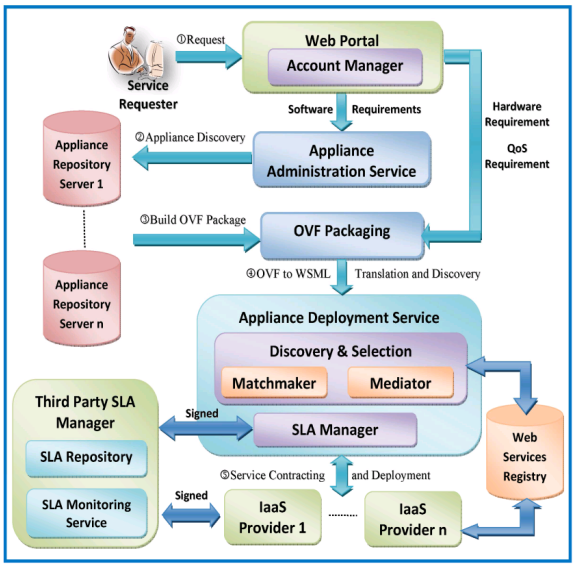

Figura 3.2: L'architettura proposta in [DTB10]

Un approccio opposto è quello proposto in [DTB10]: qui gli autori studiano il deployment di immagini (o *appliance*) nel cloud e sostengono che l'idea di avere un matching a livello di attributi è troppo poco flessibile per un ambiente così eterogeneo. Si propone quindi di sfruttare per la modellazione uno standard come la Web Service Modeling Ontology (WSMO). Il processo di deployment consiste in cinque fasi, visibili in figura 3.2: per prima cosa l'utente specifica i suoi requisiti e sulla base di essi vengono selezionate le immagini compatibili dal catalogo; poi viene costruito un package in formato standard OVF che contiene i metadati necessari. Tali metadati vengono poi convertiti in un linguaggio formale, il Web Service Modeling Language (WSML) per il matchmaking sulla base di

[1] Alex Amies, "Searching for Images in the Asset Catalog" (available for SCE registered users)

ontologie: viene quindi scelto il provider IaaS che garantisce valori di SLA e QoS richiesti e su di esso verrà migrata l'immagine in questione.

3.2.1.4 Migrazione delle configurazioni

Al momento della migrazione vera e propria, è necessario affrontare il problema di migrare ed eventualmente correggere le configurazioni degli applicativi e dei sistemi operativi, sia nel caso di una migrazione completa di una macchina virtuale, sia, a maggior ragione, di una migrazione tramite reinstallazione. In [LLM+08] viene proposto uno strumento del genere che, nel caso di sistemi operativi di tipo Unix, sia in grado di effettuare una discovery delle configurazioni di servizi esistenti, popolando dei modelli basati sul Common Information Model (CIM)[DMT11], uno standard aperto per descrivere gli elementi di un ambiente IT e le relazioni tra di essi. Sulla base di questi modelli e di appositi file di configurazione, il tool proposto permette di migrare le configurazioni tra ambienti diversi, mantenendo quindi, per quanto possibile, le funzionalità senza richiedere intervento manuale. L'altro punto affrontato dallo studio è la proposta di un algoritmo che esplora i vari servizi cercando le relazioni tra di essi, così da aiutare la tracciabilità del processo di migrazione e migliorare l'individuazione degli errori.

3.2.1.5 Migrazione

Oltre agli studi su fasi specifiche della migrazione analizzati nelle sottosezioni precedenti, sono anche stati individuati dei casi interessanti che, similmente al nostro approccio, prevedono una visione generale in grado di coprire l'intero processo.

[HSS+10] presenta un interessante studio che si sofferma in particolare sullo scenario del cloud ibrido, che è tra quelli più frequentemente richiesti in ambito enterprise. Le sfide che, secondo gli autori, sono collegate con questo scenario sono principalmente due, visibili anche in figura 3.3:

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

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

I fattori presi in considerazione sono quindi i tempi di risposta (cercando di mantenerli più bassi possibili), la data privacy (per cui i dati sensibili vanno mantenuti on-premise, puntando a conservare nello stesso luogo sia i database che le macchine che vi accedono, così che i tempi di risposta rimangano bassi). Si cerca infine di avere un risparmio sui costi il più alto possibile, modellando sia i benefici portati dalla migrazione che i nuovi costi, siano essi effettivi (per esempio un aumento dei costi di comunicazione) oppure derivati (per esempio l'aumento dei tempi di risposta nelle transazioni). La seconda parte della trattazione si sofferma su un algoritmo per la migrazione delle politiche di sicurezza, cercando di minimizzare il traffico non voluto, ma mantenendo i vincoli del sistema di origine.

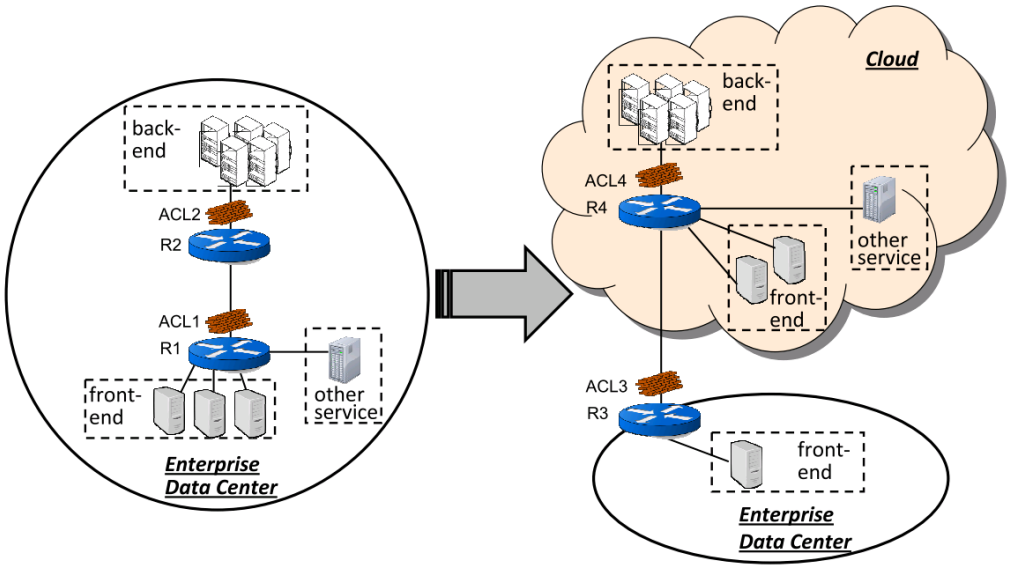

Figura 3.3: Un tipico scenario di migrazione verso un cloud ibrido

Arrivando ai punti più rilevanti per questo lavoro di tesi, troviamo lo studio di [WAB+10]; qui si sostiene come i due aspetti che più influenzano la valutazione della migrazione siano costi e rischi connessi. La migrazione viene vista come un percorso completo che consiste in sei fasi: *discovery*, nella quale vengono esplorati i sistemi originari tramite tool appositi; *analysis* e *design*, nella quale si selezionano i workload destinati alla migrazione e si valutano anche i costi, cercando un Return of Investment (ROI) positivo; *map*, in cui si prepara un file che descrive precisamente le origini e le destinazioni e che verranno utilizzate nella successiva fase di provisioning, nella quale si creano e configurano le immagini di desti-

nazione. C'è poi la fase di *migrate*, in cui si migrano le configurazioni delle applicazioni, correggendo eventuali inconsistenze; si termina con la fase di *remediate and test*, in cui si fanno le verifiche del caso. Vista la grande varietà delle piattaforme di destinazione, gli autori propongono un framework di nome Darwin che contiene al suo interno vari componenti dedicati alle singole fasi e che sono eventualmente anche utilizzabili separatamente. Concludendo, vengono segnalati una serie di problemi riscontrati nella fase di test, fra i quali il problema della mancanza di API standard sul cloud, mancanza di strumenti integrati per il calcolo del ROI e limiti alla configurabilità dell'ambiente di destinazione, sui quali gli autori intendono lavorare in futuro.

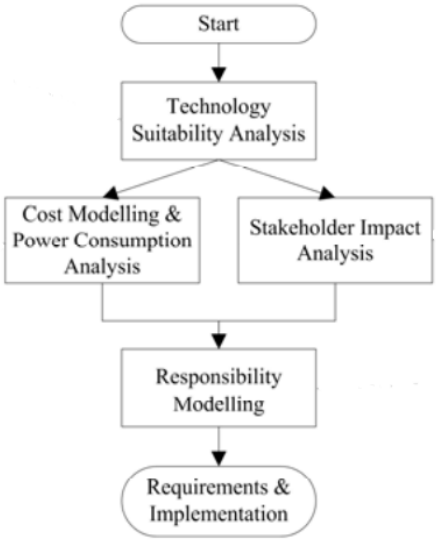

Figura 3.4: Cloud Adoption Toolkit

Due interessanti documenti sulle tematiche della migrazione sono rappresentati da [KHSBT, GKHSS10]. L'idea generale degli autori, con cui concordiamo, è quella di proporre solo un framework concettuale, vista l'eterogeneità dell'argomento, all'interno del quale si possa pensare di inquadrare una serie di strumenti che, in base alle singole esigenze, possano essere modificati o aggiunti. Tali strumenti, visibili in figura 3.4, vengono suddivisi in cinque categorie: strumenti destinati all'analisi tecnica di fattibilità, modellazione dei costi, analisi dei consumi energetici, analisi dell'impatto sugli azionisti e infine modellazione della responsabilità; gli ultimi due in particolare si soffermano sui cambiamenti organizzativi e operazionali provocati da una migrazione. La maggior parte degli

strumenti descritti sono costituiti da fogli di calcolo contenenti una serie di domande in modo da riuscire a valutare benefici e rischi, mentre uno sviluppo maggiore c'è stato dal punto di vista della modellazione dei costi, per i quali vengono appositamente estesi i diagrammi UML di deployment, con l'obiettivo di modellare il pricing a consumo tipico del cloud. Gli stessi autori propongono poi in [KHGS10] un case study pratico, che riguarda un'azienda britannica del settore petrolifero, per il quale viene applicato il framework appena descritto. L'interessante conclusione del lavoro spiega come, anche se in base ai risultati dello studio si sarebbe potuto ottenere un risparmio sui costi in caso di passaggio a un provider cloud (Amazon AWS) pari a circa il 37% nell'arco di cinque anni, quindi non trascurabile, i rischi e i cambiamenti operazionali hanno in ogni caso fatto propendere i dirigenti dell'azienda verso la decisione di non effettuare la migrazione in cloud, in quanto i rischi e gli svantaggi sono stati considerati eccessivi.

3.2.1.6 Test delle prestazioni

Il settore dei test prestazionali è molto vasto e ha diverse ramificazioni a seconda di quali elementi si vogliono verificare; concentrando, in particolare, l'attenzione sul settore che stiamo analizzando, quello del cloud, possiamo avere due famiglie di test:

1. infrastrutturali
2. applicative.

La prima categoria si pone come obiettivo il valutare le capacità computazionali del cloud, cercando di ottenere valori riproducibili e permettere il confronto tra soluzioni diverse. Uno degli studi che affrontano questa tema è [LML+11], nel quale gli autori, sulla base di una serie di test, sono in grado di confermare l'impredicibilità delle prestazioni di una macchina virtuale, che dipendono in larga misura dalla macchina fisica sulla quale essa viene creata; altri parametri determinabili invece dall'utente, come il datacenter e il periodo della giornata di esecuzione, hanno un'influenza decisamente minore.

Il secondo punto di vista è quello applicativo: infatti, al di là delle pure capacità computazionali, utili soprattutto per attività CPU-*intensive*, nella maggior parte dei casi si possono ottenere dei dati più utili dal confronto delle prestazioni su applicazioni reali, in particolare quelle web based, che si adattano molto bene al paradigma cloud. Benchmark di questo tipo esistono da parecchio tempo e fra i più diffusi ci sono TPC-W [Men02] con

il suo successore TPC-E, SpecWeb2009 e RUBiS [RUB], che simula un sito di aste simile a eBay.com. Tuttavia in [SSS+05] si sostiene come tali progetti siano limitati ai fini della ricerca sia per motivazioni economiche (sono progetti di tipo commerciale) sia perché non riescono a replicare i workload più moderni, caratterizzati da molteplici interazioni tra gli utenti e un'esperienza di navigazione molto più ricca. Gli autori propongono quindi Cloudstone, un benchmark composto da un'applicazione di esempio che segue le idee del Web 2.0 e implementata in più linguaggi e un generatore di carico, in grado di riprodurre pattern di richieste il più possibile realistici e aderenti al comportamento di un utente. Vengono poi presentati i risultati dei testi eseguiti sul cloud Amazon, dove, come metrica, viene proposto il costo mensile di ogni utente al quale fornire il servizio, così da sfruttare l'idea di pay-per-use che è tra i punti di forza del cloud.

Un'ulteriore estensione viene proposta in [CUWS11], dove gli autori propongono BenchLab, un framework modulare capace di fornire risultati significativi. Le applicazioni simulate sono diverse, una delle quali è la stessa alla base di Cloudstone; oltre a ciò, viene però posto l'accento su un componente sviluppato per registrare i comportamenti degli utenti, così da riuscire ad avere dei pattern riutilizzabili. Essi sono poi riprodotti da un componente che è in grado di simulare dei browser, compresi i tempi di attesa. L'ultimo contributo dello studio è quello di pacchettizzare i diversi elementi in modo da poterli distribuire rapidamente su macchine in luoghi differenti e poter così fare anche un'analisi sulla latenza degli accessi, per poter trovare il datacenter in grado di fornire le migliori prestazioni.

3.2.2 Soluzioni industriali

Dal punto di vista aziendale, esiste un buon numero di prodotti proprietari che si occupano del tema della migrazione enterprise: li vediamo di seguito in maniera separata e piuttosto rapida, visto che in genere i dettagli tecnici che più ci interessano non vengono resi pubblici dai produttori.

La maggior parte delle aziende più grandi del settore offre a questo riguardo tool specifici; per fare un esempio, è disponibile gratuitamente il Microsoft Assessment and Planning Toolkit (MAP)[2], che è uno strumento generico di analisi della migrazione di sistemi operativi sempre più orientato nelle ultime versioni verso scenari cloud, quali la piattaforma

[2]http://technet.microsoft.com/en-us/library/bb977556.aspx

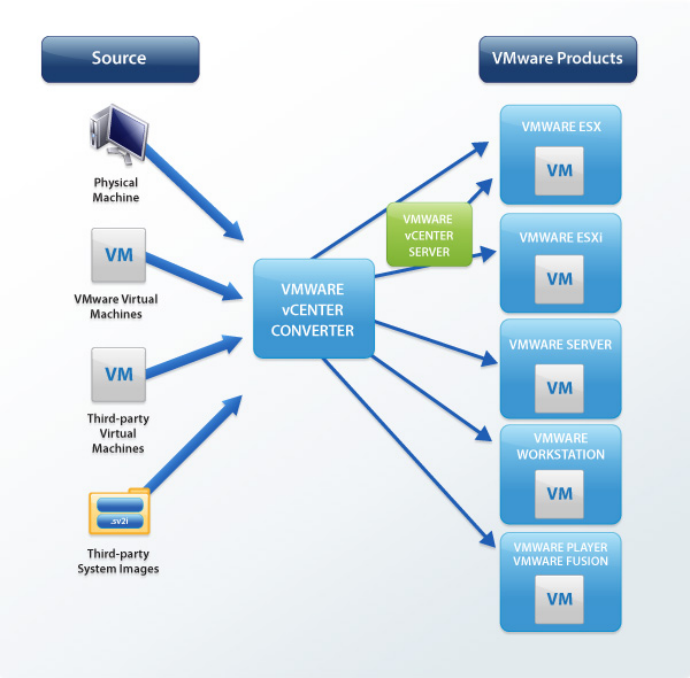

Figura 3.5: Schema di alto livello di VMware vConverter

Azure e quella SaaS Office365; in questo caso per la fase iniziale di discovery si sfruttano le informazioni raccolte tramite Windows Management Interface e, una volta raccolti i dati, viene prodotta una serie di report e vengono forniti suggerimenti per la migrazione. Il target ideale di questo ambiente sono i sistemi operativi Microsoft; è interessante notare come questa soluzione preveda l'uso di un database per conservare i dati, un'idea che riprenderemo in fase progettuale.

Uno dei grandi *player* del settore, VMware, sfrutta invece le sue tecnologie di virtualizzazione all'interno di vCenter Converter, un software che permette una conversione rapida sia partendo da server fisici che da immagini già virtualizzate in formati differenti; al termine, è possibile utilizzare le immagini virtuali generate in uno qualsiasi dei prodotti dell'azienda americana, da quelli gratuiti e studiati per singoli utenti VMware Player agli hypervisor di livello enterprise, come VMware ESXi. In questo strumento, però, l'accento è più sulla virtualizzazione di un server fisico completo piuttosto che sul cloud, per cui non ci sono al momento opzioni per l'analisi sulla fattibilità, né vi è la possibilità di scegliere la migrazione di un unico workload, come invece è possibile presso altri produttori.

I produttori minori, invece, vedono l'analisi e l'esecuzione della migrazione verso il

cloud come servizi, quindi hanno in genere delle *best practice* da seguire ma non tool veri e propri. Per esempio, OpenCrowd offre un servizio simile[3] orientato in particolare alla migrazione di software esistente che si organizza in tre fasi: Plan, con analisi di configurazioni, costi e operazioni, Port e infine Launch.

3.2.3 Temi correlati

Un tema correlato è quello della *server consolidation*: con questo termine si intende l'approccio che ha l'obiettivo di migliorare l'utilizzazione dei server (fisici), cercando di distribuire i vari workload nel miglior modo possibile. È legato alla tematica del cloud in quanto solitamente tali workload sono macchine virtuali, che sono alla base anche del cloud. Quello che il cloud aggiunge è però un'elevata dinamicità, mentre in genere la server consolidation parte dal presupposto della conoscenza storica delle variazioni di carico; questa difficoltà è bilanciata dal fatto che nel caso del cloud, in particolare quello pubblico, si mettono a disposizione dei tagli fissati di macchine virtuali, rendendo quindi la redistribuzione più semplice. Un interessante studio a questo riguardo è quello in [SB10], dove viene proposto un algoritmo di ottimizzazione per il consolidamento dei vari workload.

Simile alla server consolidation c'è il tema dello *storage consolidation*: con questo termine si intende la centralizzazione e la redistribuzione delle risorse di storage di backend in modo sia da ottimizzare il grado di utilizzo che da avere miglioramenti in termini di QoS. In [ZRE+11] si fa notare come il problema sia stato affrontato spesso in letteratura, ma nell'ottica del cloud un fattore fondamentale è l'alta variabilità dei workload e degli utenti presenti, che costringono a tenere in considerazione anche le caratteristiche temporali dei singoli workload. Per questo gli autori propongono un algoritmo che individua dinamicamente dei cluster contenenti workload complementari a quelli conflittuali, in modo da poterli tenere uniti nella fase di migrazione su diversi volumi.

In [WGR+09] gli autori si soffermano su una problematica particolare dal lato enterprise, quella della connessione, proponendo la soluzione CloudNet in grado di unire le VPN con il cloud computing. Nello specifico, gli obiettivi sono quelli di avere connessioni trasparenti con il cloud senza necessità di riconfigurazione, garanzie sulla sicurezza, isolamento delle risorse in cloud e infine provisioning rapido di risorse di rete: cambiare una caratteristica della rete deve poter essere fattibile in modo rapido. Questo permette

[3] http://www.opencrowd.com/services/migration.php

una serie di attività innovative, fra le quali la migrazione live di VM non solo all'interno di una LAN, come è possibile fare finora, ma anche in una WAN, grazie alla riconfigurazione automatica della rete. Uno scenario di questo tipo può avere una notevole importanza per il tema dell'High Availability: infatti, nell'eventualità di problemi hardware, si presenta la possibilità di migrare le macchine su un nuovo server fisico senza interruzioni al servizio.

Come si è già detto, le migrazioni live sono possibili in tutte le tecnologie di virtualizzazione disponibili sul mercato all'interno di una LAN, per esempio di un datacenter. Ciò che è più complesso sono le strategie per rilevare quando far scattare una migrazione e le modalità di scelta della destinazione migliore, oltre che le quantità di risorse da allocare per la nuova macchina virtuale. In [WSVY07] si propongono degli algoritmi per questo scopo, prendendo in considerazione due scenari: quello in cui si ha la possibilità di installare strumenti di monitoraggio nelle diverse VM e uno scenario più generico (ma ovviamente meno preciso) che si basa esclusivamente su una visione esterna, a black box, delle macchine virtuali; è possibile anche configurare una serie di parametri per poter affinare, per esempio, i tempi di reazione del sistema al rilevamento di un sovraccarico.

3.3 Confronti e considerazioni conclusive

Abbiamo dunque discusso le principali soluzioni esistenti a riguardo del tema della migrazione; a nostro giudizio, la trattazione dal lato consumer e a livello di Software-as-a-Service è sufficientemente ampia e ricca di *use case* pratici, mentre il lato infrastrutturale non è stato ancora approfondito a dovere, anche se a livello industriale sono stati senza dubbio sviluppate soluzioni che coprono questo ambito, delle quali purtroppo non si conoscono i dettagli precisi.

Le esperienze di migrazione di software sono state presentate rapidamente in quanto piuttosto specifiche e caratterizzate da notevole eterogeneità di parametri (architettura, linguaggi utilizzati, ecc.), per cui risulta difficile trarre spunti generali per affrontare questo tema. Dal nostro punto di vista, una migrazione di un software all'ambiente cloud richiede un attento studio della struttura del programma, affiancato eventualmente da un'analisi di tipo economico se, come nella maggior parte dei casi, si intende adottare anche un modello di pricing a consumo.

Le esperienze di migrazione dal lato enterprise sono invece state suddivise in base alla fasi trattate: infatti, a fianco di studi generali sul tema, presentati in sezione 3.2.1.1,

ce ne sono alcuni che affrontano temi specifici quali il discovery di sistemi esistenti o la correzione delle configurazioni. La nostra trattazione si inserirà all'interno del primo filone, cercando quindi di mantenere una visione generale dell'argomento, anche se l'accento verrà posto sull'analisi della fattibilità.

Si è deciso di mantenere l'idea di framework flessibile che emerge in alcuni lavori, suddividendolo in due macrosezioni tra loro indipendenti, quella di analisi e quella di migrazione vera e propria, a loro volta ulteriormente suddivisibili in sottoparti. In questo modo pensiamo che il nostro ragionamento possa essere utilizzato anche al di fuori di questo singolo studio; tale approccio a *pipeline* ci permette inoltre di delegare a studi dettagliati, presentati poco innanzi, alcune parti specifiche, senza descriverle in dettaglio.

La nostra attenzione si concentrerà in particolare su due fasi, che secondo noi sono meno mature e affrontate in letteratura. La prima è quella dell'analisi della migrazione dal punto di vista tecnico, al fine di fornire indicazioni in tempi brevi a un analista che si occupa di questo aspetto. Si è notato, infatti, come gli studi generali esistenti si focalizzino sull'analisi degli aspetti economici e dei cambiamenti a livello organizzativo che una migrazione può provocare, trascurando gli aspetti maggiormente tecnici che costituiscono, a nostro parere, una parte importante del processo di analisi della migrazione.

La seconda fase sulla quale ci possono essere, a nostro giudizio, dei margini di sviluppo è quella relativa ai test dinamici sulle prestazioni, necessari in fase di migrazione e consigliati dagli stessi fornitori di servizi cloud; da questo punto di vista cercheremo quindi di fare delle considerazioni generali che si possano applicare a un'ampia fascia di casi. Questo genere di prove è complementare all'analisi statica, ma può essere preso in considerazione anche separatamente.

Per gli altri passi del processo di migrazione come il discovery di sistemi esistenti o la migrazione delle configurazioni, invece, le soluzioni individuate sono mature e affermate, per cui è possibile pensare di integrarle nel framework in via di sviluppo come delle *black-box*; per i dettagli si rimanda dunque alla letteratura disponibile sull'argomento e discussa dettagliatamente in precedenza.

Capitolo 4

CloMAs

In questo capitolo andremo a descrivere il progetto dello strumento CloMAs (Cloud Migration Assistant), in grado di fornire un aiuto ad un consulente di Information Technology nell'analisi della fattibilità e della convenienza di migrazione a cloud dei propri carichi applicativi o workload. Questo strumento può essere utilizzato sia per clienti che abbiano o meno affrontato il passaggio alla virtualizzazione e il cloud di destinazione potrà essere privato o pubblico.

Le aziende stanno sempre più interessandosi alle nuove funzionalità offerte dal cloud computing, che promettono sia miglioramenti organizzativi che ottimizzazione dei costi, grazie a economie di scala e miglior grado di utilizzo delle risorse a disposizione. Lo strumento CloMAs risulta dunque utile a supporto delle aziende che, dopo aver approntato una propria strategia di alto livello per l'introduzione del cloud, vogliano valutare l'affinità di loro workload applicativi nei confronti di uno o più modelli di cloud computing.

L'idea alla base dello strumento è di confrontare una modellizzazione dei workload applicativi del cliente con quelli disponibili presso i cloud di destinazione; essa avviene mediante una serie di parametri quali CPU e memoria, che descriveremo con maggior dettaglio, che si possono aggiungere o ridurre, escludendo quelli non rilevanti.

Per quanto riguarda l'ambiente di destinazione, è prevista la possibilità di analizzare i cloud forniti da differenti provider, per permettere confronti sulla base, per esempio, di convenienza o flessibilità. Lo strumento si focalizza principalmente su uno scenario di tipo IaaS, in cui si prevede di migrare i server sui quali sono ospitati i workload del cliente verso immagini del catalogo messo a disposizione dal provider; tale migrazione include anche l'intero stack software. Non viene invece preso in considerazione all'interno del tool uno

scenario di sostituzione dei vari software con immagini PaaS e SaaS rese disponibili su Cloud; in questo caso, il problema si sposta più sulla selezione delle immagini e viene ben affrontato in studi quali [DTB10].

L'utilizzo dello strumento fornisce poi delle indicazioni di Business Intelligence, in particolare il modello di Cloud più adatto alla migrazione dei workload del cliente e la complessità per la loro migrazione, così da consentire di rispondere ai principali quesiti che nascono in fase di migrazione:

- la compatibilità del workload con l'ambiente di destinazione e nello specifico la compatibilità di infrastruttura, sistema operativo e middleware
- il rispetto di requisiti operazionali e non funzionali quali prestazioni e disponibilità
- il rispetto di requisiti di sicurezza e di privacy
- i benefici che si possono ottenere in seguito alla migrazione.

Sulla base di queste informazioni si potrà partire in un secondo tempo con la fase di migrazione vera e propria, che prevede lo spostamento fisico degli ambienti e le operazioni ad esso correlate.

4.1 Parametri presi in considerazione

Sulla base dei nostri studi e dello stato dell'arte è stata selezionata una serie di parametri che possono essere presi in considerazione per l'analisi di opportunità di migrazione; quello che va tenuto sempre in considerazione è che non tutto può essere migrato con successo – in modo da ottenere benefici – nel cloud, per cui è necessario affiancare un ragionamento approfondito all'utilizzo di uno strumento automatico come quello che proponiamo. Vediamo ora in breve i parametri che sono stati considerati rilevanti nel nostro lavoro.

4.1.1 Workload

Il workload, o carico di lavoro, può essere definito in modo generale come le richieste complessive fatte dagli utenti e dalle applicazioni di un sistema.

Nel nostro caso specifico, possiamo definirlo come un carico elaborativo, modellizzato attraverso uno o più server o immagini virtuali, ognuna con i suoi attributi di potenza

computazionale, il virtualizzatore e il sistema operativo con i middleware installati. Questo tipo di modellizzazione ci permette di effettuare un confronto con i servizi IaaS come vengono messi a disposizione dai differenti provider.

Come vedremo tra poco, un workload è in relazione con le diverse immagini virtuali, per cui in un caso classico, come un'architettura a 3 *tier,* potrebbero entrare in gioco più immagini, una delle quali dedicata alla gestione dei dati, l'altra alla loro elaborazione, mentre una o più immagini saranno destinate al layer di presentazione. Questa classificazione viene spesso conservata nel sistema di configurazione aziendale, per cui può essere molto utile estrarre queste informazioni sia per avere un'idea delle relazioni tra le immagini esistenti sia per capire quali sono i migliori candidati alla migrazione. La nostra idea è che, con l'aiuto di un esperto del dominio di partenza, si cerchi di categorizzare ogni workload del sistema di origine, che spesso consiste in nomi assegnati in modo automatico, in un sottoinsieme più limitato, quale per esempio quello riportato poco oltre.

A questo punto si può assegnare una facilità di migrazione alle varie categorie individuate e tale coefficiente potrà essere utilizzato o direttamente nell'output oppure costituire uno degli elementi per il calcolo della difficoltà complessiva di migrazione come spiegheremo in sezione 4.5.2.

Una possibile categorizzazione dei workload è quella proposta da IBM, che prima vediamo in una visione generale in figura 4.1, per descriverli poi in maggior dettaglio.

In questa figura vediamo la suddivisione di workload in base alla facilità di una migrazione verso il cloud. In alto a destra ci sono i workload innovativi o più pronti per il cloud, mentre andando verso sinistra ci sono quelli meno indicati per motivi diversi, ad esempio per il rischio (reale o presunto) legato al trasporto fuori dall'azienda dei dati importanti e sensibili; un'altra categoria poco adatta alla migrazione è quella dei workload poco standardizzati, in quanto per sua natura il cloud lavora meglio in presenza di standardizzazione.

Una seconda suddivisione di alto livello è quella presentata di seguito, in dieci categorie, cui assegnare i vari workload ottenuti dall'esplorazione di un sistema:

Web Serving Workload collegati alla fornitura di contenuti Web sia statici che dinamici, allo streaming di elementi multimediali, notizie RSS, mashup, SMS, ecc.

Applicazioni Web Workload collegati ad applicazioni che espongono Web service, applicazioni di e-Commerce, e-Business, Java application server, Rich Internet Applica-

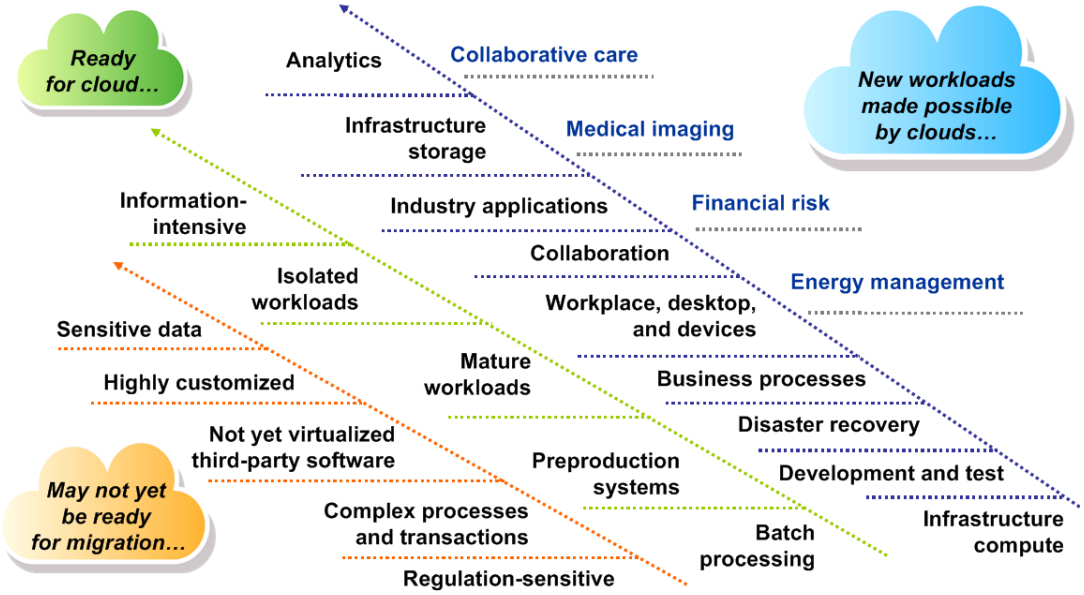

Figura 4.1: Tipologie di workload in base alla possibilità di migrazione sul cloud

tion (Adobe Flash, JavaFX, Microsoft Silverlight), motori di ricerca Web, ecc.

Business Intelligence/Data Warehouse Workload collegati al data mining, warehousing, analitica di dati in streaming (per esempio per il rilevamento di frodi), text mining, analitica competitiva, business intelligence e applicazioni per decisioni di business, ecc.

ERP/SCM Workload collegati a planning e scheduling di risorse enterprise, scheduling e planning ingegneristico e della produzione, applicazioni di gestione della catena dei fornitori, gestione degli ordini di acquisto, applicazioni finanziarie, Customer Relationship Management (CRM), applicazioni per le risorse umane, ecc.

Analitica Workload collegati a Online Analytic Processing (OLAP), business optimization, previsioni di marketing e vendite, report per il management, applicazioni di gestione e analisi del rischio, valutazione del credito, analisi del portafoglio finanziario, ecc.

Numerici e batch Workload collegati a design ingegneristico e analisi, applicazioni scientifiche, high performance computing, simulazioni numeriche tipo Montecarlo, pro-

cessamento di immagini mediche, computazioni batch intensive con dati a virgola mobile, ecc.

Collaborazione Workload collegati ad applicazioni Web 2.0 per condivisione e collaborazione online, messaggistica istantanea, mail server (SMTP), VOIP, ecc.

Archiviazione e stampa Workload collegati a stampe, file system, archiviazione e recupero

Desktop Workload collegati al desktop based computing, desktop per helpdesk e applicazioni di supporto e applicazioni di desktop management

Sviluppo e test Workload collegati a processi di test e sviluppo, gestione delle immagini, ecc.

Dalla nostra esperienza, i primi candidati per una migrazione sono gli applicativi Web, che sono scalabili anche orizzontalmente, gli ambienti di test e sviluppo, che permettono di risparmiare investimenti su macchine che poi verranno utilizzate solo per brevi periodi di tempo, gli applicativi mobili e le applicazioni batch in parallelo (per esempio quelle che sfruttano l'algoritmo MapReduce nella sua implementazione opensource Hadoop, in grado di suddividere un compito su diversi nodi per poi raccogliere e riunire i risultati prodotti).

Questo tipo di classificazione, pur essendo abbastanza complicata, in quanto in genere richiede una conoscenza approfondita dell'ambiente da migrare e un'assegnazione manuale di ogni singola immagine alla categoria più adeguata, può fornire, assieme agli altri parametri, una prima indicazione dei candidati più probabili alla migrazione.

4.1.2 Relazioni

Un elemento molto importante, soprattutto in ambito enterprise, è quello delle relazioni fra le varie immagini e i workload di cui fanno parte. Nel caso più semplice, ogni immagine può appartenere a un unico workload (relazione di tipo 1-N), mentre, in una parte dei casi di studio affrontati, è emerso come la relazione risulti più complessa (N-M). Questo, nello specifico, significa che ogni immagine può appartenere a uno o più workload, mentre ogni workload può essere costituito da uno o più immagini.

Dal punto di vista della migrazione, risulta interessante effettuare un'analisi di queste relazioni, perché può dare un'idea della complessità della migrazione. Anche se, per avere

un'idea completa della situazione, sarebbe necessario andare a vedere le configurazioni dei singoli sistemi, possiamo comunque sostenere che due immagini, se appartengono allo stesso workload, probabilmente vanno migrate o tutte o nessuna di esse, in quanto lasciarne una sull'infrastruttura precedente potrebbe creare dei problemi di latenza e di permessi. Quello che si è fatto è quindi "premiare" le immagini che hanno un minor numero di immagini ad esse collegate, in quanto sono le prime candidate alla migrazione, basandosi sulla facilità di essa. L'algoritmo 1 ci permette di individuare le relazioni tra immagini e workload in modo completo, in modo da poter avere a disposizione questa informazione per elaborazioni successive.

Va notato come, nel caso peggiore, potremmo avere che ogni immagine è collegata a tutte le altre attraverso una serie di passaggi; in questo caso, ovviamente, questo parametro non può darci risultati utili, in quanto la difficoltà sarebbe la stessa per tutte le immagini di origine e andrà quindi trascurato.

4.1.3 Taglio delle immagini

Con taglio delle immagini si intendono i parametri che l'utente può scegliere al momento della creazione di un'istanza; una delle caratteristiche del cloud è infatti quella della standardizzazione, per cui è possibile scegliere solo tra tagli predeterminati. In genere, i produttori mettono a disposizione dei tagli che si differenziano per caratteristiche fisiche quali numero di CPU virtuali disponibili, quantità di RAM e storage incluso a livello di istanza. Ad esempio, nelle soluzioni IBM analizzate, SCE ha immagini con tagli fissi, mentre SCE+ permette una maggior flessibilità e le varie caratteristiche possono essere diminuite o accresciute fino alla massima dimensione consentita dai server fisici.

Questa è una delle parti più importanti, ma anche tra le più studiate in letteratura, in quanto sta alla base della migrazione verso un qualsiasi ambiente di cloud pubblico. In questo caso la ricerca che va fatta è quella dell'immagine che più si avvicina al taglio fisico di quella di origine, anche se essa può solo dare un'idea di base dell'ambiente di destinazione, utile per capire i costi più che un dimensionamento preciso. Infatti, sia Amazon sia IBM danno delle indicazioni generali sulle prestazioni dei componenti serviti: per esempio, nel caso delle CPU, si parla di CPU virtuale ad una certa frequenza, ma ovviamente questo non basta per capire le prestazioni in un ambiente reale, che viene influenzato in maniera più o meno marcata dalla gestione delle risorse fisiche che viene effettuata dall'hypervisor.

Algoritmo 1 Algoritmo per ottenere per ogni workload quelli ad esso collegati

INPUT
1)DICTIONARY<WORKLOAD, LIST<IMAGE>> INPUT1
2)DICTIONARY<IMAGE, LIST<WORKLOAD>> INPUT2
OUTPUT
FOR EACH WORKLOAD, A LIST OF THE CONNECTED WORKLOADS; SINCE IT'S SYMMETRIC, THE SAME HOLDS FOR IMAGES

```
//Start from workload list
foreach (workload in Lista1) {
  temp_images = input1.get(workload); //images connected to the current workload
  workload_ok += workload; //list of visited workloads
  foreach (image in temp_images) {
    //check if not already visited
    if (image is not in image_ok) {
      image_ok += image;
      temp_workloads = input2.get(image);
      foreach (workload2 in temp_workloads) {
        if (workload2 is not in workload_ok) {
          temp_images += input1.get(workload2);
          workload_ok += workload2;
        }
      }
    }
  }
  //workload_ok is a list of workloads connected to the current one
}
```

Mentre per uso interno esistono strumenti che possono aiutare il dimensionamento, gli stessi produttori suggeriscono agli utenti finali di fare dei test delle prestazioni dei propri sistemi e adeguare poi la configurazione prescelta, scegliendo se necessario un taglio superiore. Un'altra strategia possibile, che vedremo in modo pratico nel capitolo successivo, consiste nel partire sempre dall'immagine meno costosa all'interno della combinazione di sistema operativo e parallelismo necessario (32 o 64 bit) e poi fare dei test, istanziando tagli superiori se ci si accorge che quello attuale non è sufficiente. Un approccio di questo tipo, invece, era difficilmente realizzabile in un'infrastruttura tradizionale non virtualizzata.

Oltre al puro taglio fisico delle immagini, in questa fase viene preso in considerazione anche il sistema operativo richiesto, per cui, se c'è la necessità di avere dei sistemi operativi particolari, sarà possibile scartare da subito i provider che non li supportano.

4.1.4 Software e middleware installato

Nei sistemi di origine si avrà una lista di software e middleware che sono installati nel sistema; tale lista potrebbe eventualmente essere ripulita in maniera più o meno automatica dai software non rilevanti per la migrazione, quali per esempio utility di compressione o prodotti della famiglia Microsoft Office estranei all'utilizzo della macchina come workstation.

Tutti quelli rimanenti, nel caso di una migrazione per reinstallazione, andranno reinstallati, previa verifica di una serie di parametri, che elenchiamo di seguito:

licenze è necessario verificare per ogni software la sua compatibilità, in termini legali, con l'ambiente di destinazione. Alcuni produttori, infatti, vendono licenze che sono basate sul numero di core fisici, dei quali un utente di servizio Cloud chiaramente non ha visione; questo tipo di licenza è dunque un limite alla possilibità di usare un servizio di tipo IaaS. Altri produttori addirittura proibiscono completamente l'installazione sul cloud e quindi questo potrebbe essere un parametro che impedisce di migrare nel cloud l'immagine che contiene tali software

SO si deve anche verificare, nel caso di una migrazione di sistema operativo, che la nuova configurazione sia compatibile con il software in questione; potrebbe infatti essere necessario o l'acquisto di una nuova versione del software oppure una sua riscrittura, nel caso esso fosse stato sviluppato appositamente per l'azienda. Tutti questi problemi potrebbero avere un costo per l'azienda e quindi far sconsigliare la migrazione per le immagini contenenti quel software.

Questi due elementi sono legati ad ogni versione di un software e si possono riassumere con un coefficiente di sforzo nella migrazione del cloud, che indica la complessità richiesta e che può essere combinata con il coefficiente di sforzo richiesto dalla migrazione a un sistema operativo diverso. Va fatto notare come, in questo momento, entrino anche in campo le relazioni all'interno del dominio, che noi modelliamo come workload: se quindi verifichiamo che un'immagine sfrutta un database che non può essere migrato per motivi di licenza in cloud, ciò impedirà la migrazione dell'intero workload.

Ricordiamo come, nella nostra analisi, si parta da un punto di vista infrastrutturale, per cui l'elemento base è l'immagine virtuale più che il singolo software; nel caso di un

approccio di tipo SaaS, invece, andrebbe posto l'accento anche sulla struttura del software da migrare in cloud.

4.1.5 Storage

In un datacenter interno, generalmente si hanno configurazioni dello storage avanzate e stratificatesi nel corso degli anni, nelle quali esso viene condiviso tra più macchine con la possibilità di partizionamento dinamico su determinate piattaforme, in particolare quella AIX. In un ambiente cloud, al momento, la visione dal punto di vista dell'utente è molto più semplificata, per cui, nel corso dell'analisi, faremo solo delle considerazioni sullo spazio complessivo, senza entrare nei dettagli fisici di realizzazione.

Di base ogni provider fornisce uno storage interno alle immagini virtuali, legato al ciclo di vita delle macchine virtuali create e uno esterno, che offre la persistenza e può essere montato su differenti istanze anche se non contemporaneamente. Alcuni provider cloud, per esempio Amazon con Simple Storage Service (S3) e IBM, attraverso il partner Nirvanix, offrono anche un secondo tipo di storage persistente, che si caratterizza per la scalabilità e che viene acceduto tramite apposite API piuttosto che essere visto in maniera classica come un'unità disco. Quest'ultimo tipo di storage offre molti vantaggi, ma richiede d'altra parte un ripensamento dell'architettura del sistema.

Dal punto di vista della nostra analisi, prenderemo come input lo storage disponibile come restituito dagli strumenti di discovery e si potrà pensare di cercare dal lato del cloud una corrispondenza con lo spazio offerto insieme a un'istanza. Tuttavia vista la possibilità di poter comunque ampliare lo spazio disponibile attraverso lo storage persistente che raggiunge ormai dimensioni considerevoli (nell'ordine dei terabyte), si può pensare, senza timore di grossi errori, di trascurare questo parametro dal punto di vista dell'analisi, mentre, probabilmente, sarà necessario effettuare delle modifiche al momento della migrazione per correggere problemi legati ai diversi sistemi di accesso allo storage.

Un punto aperto, che potrebbe essere ulteriormente approfondito, è quello di come distribuire lo storage: infatti, come si è detto, le offerte di storage hanno caratteristiche molto diverse per quanto riguarda sia costi che prestazioni e affidabilità. Per questo motivo si potrebbe pensare di eseguire un'ulteriore analisi più specifica in questo campo, in modo da ottenere la combinazione migliore che rispecchi la configurazione esistente o quella desiderata dall'utente.

4.1.6 Parametri legati al provider

Questa sezione riguarda una serie di parametri generali che dipendono dai provider che forniscono il servizio cloud e possono influenzare la scelta. Nel nostro studio ne mettiamo in evidenza due: posizione dei datacenter e SLA garantito.

Per quanto riguarda il primo punto, mentre a livello consumer l'indeterminatezza della locazione precisa dei datacenter non rappresenta un grande problema, a livello enterprise ciò è piuttosto sentito per una serie di motivi. Fra questi, ad esempio, la legislazione, che in certi paesi potrebbe essere difforme da quella del paese in cui ha sede l'azienda, in particolare per quanto riguarda la riservatezza dei dati: infatti, prima di distribuire i dati su un server esterno, l'azienda vorrebbe avere garanzie sulla protezione di tali informazioni, garanzie che spesso non è possibile avere in alcuni stati. Per questo si è deciso di associare a ogni provider i datacenter esistenti, con la loro locazione e il paese di riferimento, perché questo può essere un fattore in grado di influire sulla scelta del fornitore a cui affidarsi e qualora questo, come IBM, disponga di più datacenter, sulla decisione di quale di questi utilizzare.

Il secondo punto è quello del SLA, il quale tuttavia, nel nostro modello, verrà considerato come un'opzione del singolo taglio delle immagini in modo da coprire il caso dell'SCE+, presentato in sezione 2.5, dove sono possibili più opzioni a questo riguardo.

4.1.7 Altri parametri dinamici

I parametri dinamici sono molto importanti in vista di una migrazione, ma sono più difficili da collezionare.

Relativamente alla capacità computazionale, spesso i clienti non hanno informazioni storiche sul comportamento prestazionale e sull'utilizzo delle risorse dell'ambiente di partenza e d'altra parte i provider non forniscono statistiche e benchmark delle prestazioni degli ambienti cloud. È possibile, facendo richiesta ai fornitori di servizi, avere indicazioni sui modelli hardware sui quali vengono ospitate le immagini, ma non sono disponibili indicazioni sulle modalità di distribuzione delle VM sulle macchine fisiche, per cui risulta molto difficile sapere quale sia la potenza elaborativa erogata dai vari tagli offerti nel catalogo di un provider IaaS.

A proposito della connettività, in ambiente enterprise interni o anche quelli dati in outsourcing, esistono dei parametri ben precisi, riportati anche nei contratti che riguardano

l'ampiezza di banda garantita, con conseguenti penali da pagare in caso di problemi. I provider cloud, invece, danno in genere delle informazioni piuttosto generiche a questo riguardo, valide più a livello dell'intero datacenter che non a livello di istanza. Per fare un esempio pratico, l'IBM, per il suo datacenter SCE di Ehningen, situato nelle vicinanze di Stoccarda, sostiene come la latenza di accesso dall'Italia e dall'Europa in genere sia bassa visto il posizionamento sulla dorsale di collegamento europea, ma invita comunque a fare dei test specifici. Ogni macchina del datacenter ha 1 gigabit di connessione per l'accesso alle istanze e 1 gigabit dedicato al management, ma, in assenza di informazioni precise sul numero di istanze ospitate su ogni macchina, non è chiaro di quanta banda sia possibile disporre in un utilizzo normale.

Per questi motivi non è possibile valutare questi parametri a livello statico, quindi a livello della nostra analisi, ma gli stessi fornitori di ambienti cloud suggeriscono di effettuare test dinamici su delle configurazioni pilota, sulle quali non ci soffermeremo.

4.2 Parametri trascurati

Alcuni parametri non verranno presi in considerazione nella nostra trattazione; in particolare, non verranno affrontati i temi che toccano i cambiamenti organizzativi (per esempio l'evoluzione del ruolo del personale IT di un'azienda, che si troverà ad affrontare una serie di compiti differenti, anche a livello di supporto che possono fornire agli utenti finali del servizio) né quelli più propriamente decisionali (legati, per esempio, alla valutazione dei rischi maggiori riguardanti la protezione dei dati rispetto a una struttura interna ormai consolidata), che sono una caratteristica della migrazione enterprise. Se infatti, in ambito consumer, l'idea del cloud ha successo anche in presenza di indeterminatezza sulla locazione vera e propria dei dati, nel settore in questione la localizzazione dei dati è di importanza primaria, sia perché essi vengono considerati una delle proprietà fondamentali di un'azienda sia per motivi puramente legali o connessi al pagamento di tasse e simili.

Anche l'aspetto dei costi viene affrontato marginalmente; esistono infatti strumenti specializzati in grado di impostare un documento contenente il Return of Investment (ROI) previsto per la migrazione, con i risparmi sui costi operativi e confronti dettagliati sulla convenienza economica.

4.3 Modello dei dati

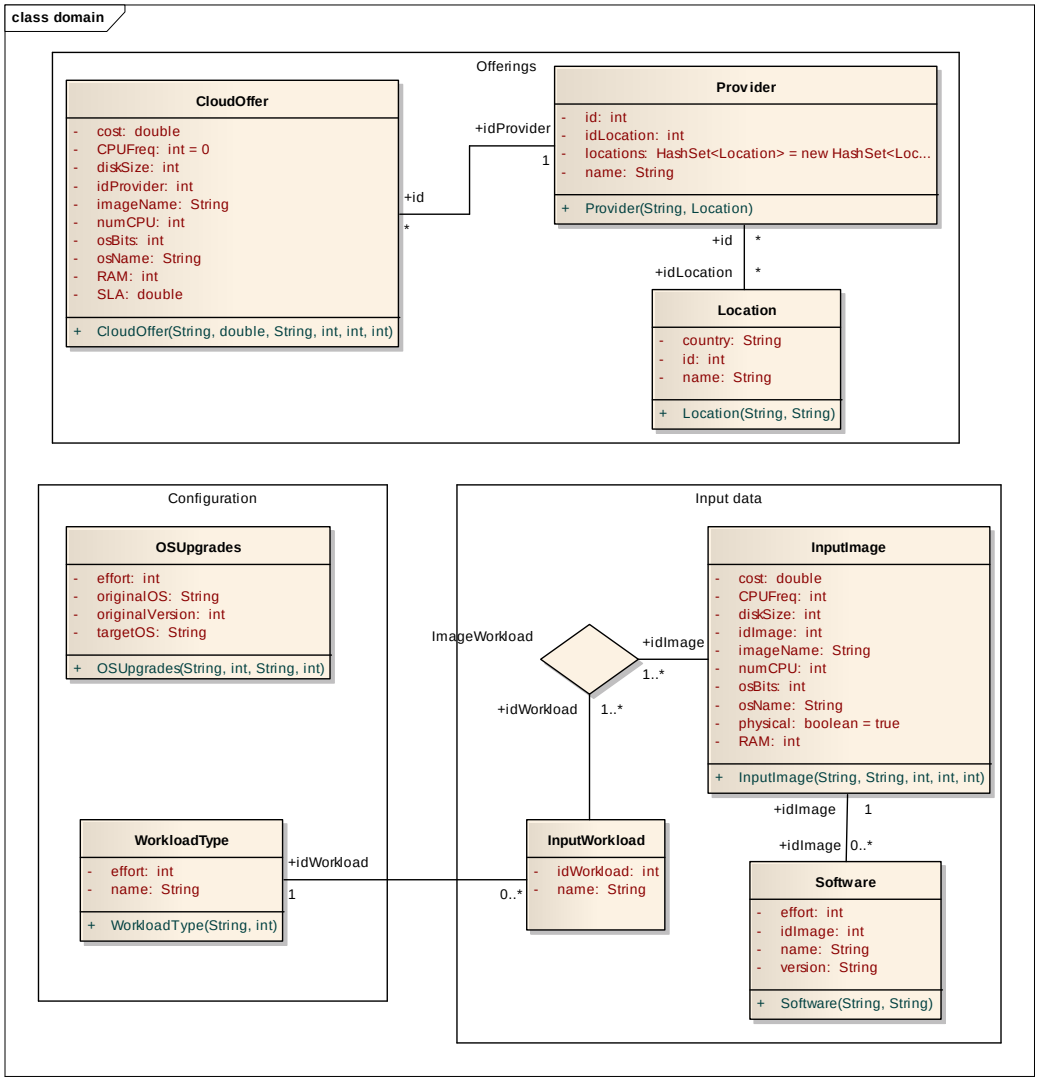

Figura 4.2: Diagramma UML del modello del dominio

Sulla base delle considerazioni appena fatte riguardo ai parametri da considerare, si può derivare un primo modello dei dati, riassunto nel diagramma UML in figura 4.2. Possiamo notare la presenza di una parte legata al provider, che include le locazioni dei datacenter offerti; troviamo poi le offerte proposte da ogni provider, che include anche lo SLA garantito per ognuna di esse.

La seconda parte è quella legata all'ambiente di origine: in questo caso l'informazione principale è sicuramente contenuta nella classe delle immagini di input, dove troviamo i parametri legati a numero e frequenza delle CPU, dimensione dello storage e molte altre informazioni più o meno utili a seconda delle singole esigenze, fra le quali se un immagine è già stata virtualizzata oppure corrisponde a una macchina fisica, per cui sarà necessario virtualizzarla attraverso appositi strumenti prima di effettuare la migrazione in ambiente cloud. Oltre a questi troviamo le parti dedicate ai singoli workload, con il mapping verso i workload standard definiti a livello globale, definiti anch'essi in una classe a parte e infine le relazioni con le immagini.

Infine, l'ultima grande sezione presente, sempre riferita ai dati in input, è quella relativa ai software rilevati su ogni immagine, con la loro versione e la complessità per la migrazione al cloud. Per avere una maggior precisione si potrebbe pensare di avere un parametro di compatibilità per ogni singolo sistema operativo, in modo da poter specificare con maggior dettaglio gli ostacoli alla migrazione posti dal software. Tuttavia, dato che nel nostro strumento il software non è il fattore principale di scelta, si è preferito considerare un parametro unico per la difficoltà di migrazione, indipendente dal sistema operativo; ciò rende anche più rapido l'ottenimento di questi dati, visto che verificare la compatibilità per ogni singolo software potrebbe essere un'operazione lunga e difficoltosa, senza precludere la possibilità di segnalare l'impossibilità di migrare in cloud un software specifico.

Troviamo poi una sezione che contiene le parti che possiamo definire di configurazione e mapping dei sistemi operativi (OSUpgrades) e che include anche i workload standard definiti a livello di progetto.

Nel diagramma possiamo vedere, infine, anche i collegamenti tra le varie classi presentate, che in alcuni casi sono da uno a molti, in altri molti a molti (in particolare nella relazione tra immagini e workload, come discusso in precedenza).

4.3.1 Informazioni minime

All'interno del modello dei dati è possibile mettere in luce le informazioni minime che sono necessarie per un processo di analisi; ovviamente, maggiore sarà il dettaglio, più saranno le possibilità di ottenere dati precisi in output.

Per quanto riguarda le immagini di input, a nostro giudizio è necessaria la presenza almeno del sistema operativo e del numero di bit, per permettere poi una verifica con

il cloud di destinazione; in loro assenza, l'immagine non verrà presa in considerazione. Anche le informazioni sui workload sono facoltative, nell'ottica di permettere un'analisi che affronti il problema senza prendere in considerazione il ruolo assunto dai workload. Infine, è possibile omettere completamente i software presenti su un'immagine, nel caso tale informazione non sia disponibile o non sia ritenuta rilevante.

4.4 Interazione con l'utente

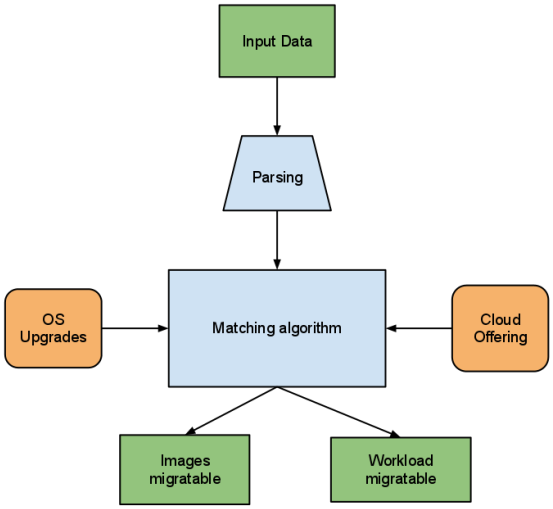

Figura 4.3: La struttura generale dell'interazione

Passiamo ora a descrivere l'interfaccia dello strumento realizzato, che è stata riassunta in figura 4.3.

4.4.1 Dati in input

I dati previsti in input sono quelli relativi ai parametri presentati in figura 4.1. Come diremo tra poco, nel nostro progetto non sono stati posti vincoli riguardo alle origini di tali dati, che potrebbero essere sia strumenti molto ricchi dal punto di vista delle opzioni come un CMDB fino a dei dati raccolti manualmente dall'analista, sia una combinazione di queste due possibilità. Per questo non sempre essi saranno completi e anzi, nella maggior parte dei casi, si avrà solo una parte dei parametri esposti poco fa. Ovviamente i risultati dell'analisi

andranno calibrati in base ai dati ottenuti e probabilmente si otterranno indicazioni meno precise in assenza di dati sui sistemi in origine.

Se, per esempio, non avessimo il numero di CPU presenti sulle immagini da migrare, non sarebbe possibile prendere in considerazione tale parametro e le immagini sul cloud di destinazione verranno selezionate solo in base agli altri parametri e quindi con minor precisione. Questo implicherà un impegno e un dispendio di tempo maggiore nella fase di test, in quanto andrà verificato che le immagini prescelte siano effettivamente adeguate alle esigenze del sistema; la presenza di molti parametri in ingresso, invece, consentirà di calibrare le immagini di destinazione con maggior precisione, riducendo dunque la fase di test, anche se essa andrà in ogni caso prevista per la variabilità delle prestazioni tipica dell'ambiente cloud.

Oltre ai dati caratteristici dell'ambiente di origine, c'è una serie di parametri di configurazione che vengono inseriti con le stesse modalità e rappresentati nel nostro dominio tramite due classi. Il primo di essi consiste in una tripletta con sistema operativo di origine, sistema di destinazione disponibile sul cloud e difficoltà richiesta da tale migrazione. Tale difficoltà potrebbe essere suddivisa su tre livelli:

- bassa o nulla, se il sistema operativo rimane lo stesso, in quanto le operazioni richieste saranno poche

- media, se è necessario un avanzamento di versione (ad esempio da Windows Server 2003 a 2008)

- alta, nel caso di cambiamento radicale dell'architettura (per esempio il porting da AIX o un altro UNIX a una distribuzione Linux), la più complicata perché richiede un porting completo delle applicazioni specifiche presenti sul sistema con possibili riscritture di software o ricerca e passaggio a soluzioni equivalenti.

Il secondo parametro generico di input è quello delle informazioni sulle offerte disponibili sul cloud e quelle legate ai provider come i datacenter che essi mettono a disposizione; tali dati sono facoltativi, ma possono rappresentare criteri di utile discrimine per l'utente finale dello strumento in corso di sviluppo.

4.4.2 Dati in output

Per quanto riguarda l'output previsto, esso consiste, in definitiva, in una lista che contenga le immagini che, in base ai parametri presi in considerazione, possono essere migrate, escludendo quelle che vanno invece scartate dall'operazione; tale lista può eventualmente essere ordinata in base a considerazione su singoli parametri (per esempio il costo) o su parametri combinati.

Volutamente, si è deciso di lasciare un ampio margine di scelta all'utente nella scelta della conformazione dell'output: un output predeterminato ha infatti dei limiti sia per i dati disponibili in input, spesso assenti o parzialmente specificati, sia per le esigenze dei singoli casi: per questo, la nostra idea è quella di presentare all'utente un'interfaccia nella quale si possano inserire query che agiscano sul modello dei dati. Per i casi più comuni e per facilitare la vita dell'utente dello strumento, possono essere previste in anticipo delle query pronte da eseguire, ma questa scelta permette una maggior flessibilità rispetto a un output unico scelto a priori dal progettista. In generale, a partire dal nostro modello dei dati, è possibile avere due visioni complementari dell'output, in base al punto attorno al quale ruota l'analisi.

4.4.2.1 Visione dal lato delle immagini

Se prendiamo in considerazione questo punto di vista, l'output basilare dell'analisi della migrazione sarà una lista di immagini, per le quali è possibile e consigliabile effettuare la migrazione in base ai parametri previsti. Per fare un esempio, l'analista potrebbe essere interessato ad avere il numero dei software presenti sull'immagine di origine, così da avere un'idea basilare del tempo che potrebbe essere speso in questo ambito; altre informazioni utili, combinabili a piacere con le precedenti, potrebbero includere quella delle immagini per cui non è necessario l'upgrade del sistema operativo, che è un'operazione a volte complessa e costosa in termini di ore impiegate.

4.4.2.2 Visione dal lato dei workload

Analizzando la situazione partendo dai workload, si possono avere due punti di vista: quello combinato e quello indipendente. Nel primo caso un workload è preso come entità a sé stante e, se esso include un'immagine che per un motivo qualsiasi non può essere migrata, l'intero workload verrà segnato come non migrabile. Questo include anche altre immagini

che potrebbero essere migrabili in base a tutti i parametri presi in considerazione, ma non verranno migrate; se altri workload comprendevano tali immagini, verrà impedita a catena la migrazione di tutti i workload. Nella visione indipendente, invece, in un caso come quello appena descritto, è possibile pensare di sdoppiare l'immagine migrabile e lasciarne una copia nell'ambiente originare, per evitare problemi con il workload che non può essere migrato.

L'idea di spostare un workload nella sua interezza nasce in particolare da due tipi di problematiche: il primo è legato alla latenza, per esempio nel caso di un application server che deve accedere a un database nel minor tempo possibile, elemento non scontato se le due istanze fossero disposte una in cloud e una in un ambiente diverso, non raggiungibile tramite un collegamento dedicato. Il secondo tipo è quello dei parametri di sicurezza delle diverse macchine virtuali, per cui il fatto di avere macchine in due ambienti differenti costringe ad aprire delle porte e ad ulteriori configurazioni che potrebbero causare rischi nel caso di attacchi malevoli.

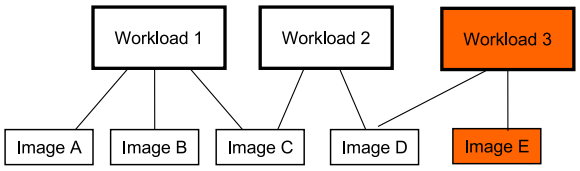

Figura 4.4: Workload combinati e indipendenti

Un esempio pratico è visibile in figura 4.4: data l'immagine E non migrabile, nel caso dei workload combinati, anche il workload 2 non potrà essere migrato perché comprende l'immagine D e a sua volta neanche per il workload 1 si potrà prendere in considerazione la migrazione, anche se tutte le immagini che lo compongono sono migrabili. Nel caso dei workload indipendenti, invece, l'immagine D verrebbe duplicata e quindi questo permetterebbe la migrazione del workload 2 e del workload 1 nel cloud.

Questa visione potrebbe essere l'unica prodotta oppure essere anche affiancata a quella precedente, risultando utile soprattutto nel caso si abbiano informazioni certe sui workload ai quali ogni immagine appartiene.

4.5 Algoritmo di matching

Vediamo ora come avviene il matching, che è il cuore del nostro sistema: si hanno in input due liste, una contenente le immagini per le quali va analizzata la possibilità di migrazione, mentre la seconda include le immagini disponibili sul cloud, che potrebbero appartenere all'offerta di uno o più provider. Viene inoltre preso in considerazione un file, elaborato dall'utente prima dell'esecuzione, che rappresenta quale deve essere il mapping tra il sistema operativo di origine e quello di destinazione.

Il primo passo dell'algoritmo 2 è quello di recuperare il sistema operativo corrispondente nell'ambiente di destinazione. Ciò non sarebbe necessario se nel cloud potessimo avere qualsiasi sistema operativo, ma la standardizzazione tipica del cloud rende necessario assegnare a ogni sistema operativo di origine uno di destinazione all'interno di una scelta più ristretta. Ricordiamo come questa limitazione sia presente, seppur in misura minore, anche in uno scenario di cloud privato, in quanto sarà comunque impossibile che il software di gestione delle macchine virtuali supporti ogni singola combinazione di sistema operativo esistente, per motivi di convenienza sia economica che tecnica. In questa fase possiamo quindi configurare il sistema operativo di destinazione e anche impedire la migrazione: se, per esempio, abbiamo un sistema di tipo Unix che non è disponibile sul cloud di destinazione, l'immagine verrà immediatamente scartata (ed eventualmente aggiunta a una lista delle immagini non migrabili) e si eviterà di scorrere tutte le possibili immagini di destinazione.

La seconda fase è l'analisi di tutte le immagini del cloud e dei relativi parametri; si parte, per esempio, dagli elementi quali il sistema operativo (e numero di bit), per passare poi alla RAM, alle dimensioni dello storage e così via. I parametri analizzati possono variare a seconda della disponibilità di dati in ingresso, ma il comportamento rimane lo stesso: i vari controlli vengono effettuati sequenzialmente e, se uno di essi fallisce, si esce dal ciclo corrente; se invece tutti i controlli sono stati passati con successo, si potrà aggiungere l'immagine di destinazione alla lista di immagini compatibili. È consigliabile iniziare il controllo dai parametri più generali, quali i bit del sistema operativo, così da poter scartare il prima possibile le immagini non adeguate.

Il terzo stadio dell'algoritmo consiste nella selezione della migliore immagine tra quelle compatibili. Nella fase precedente, infatti, otterremo molto probabilmente una serie di immagini che avranno uno stesso sistema operativo, un minimo di CPU, di RAM e così

via: la migliore di queste può essere estratta ordinando tale lista e poi prendendo il primo elemento.

Per quanto riguarda i parametri secondo i quali ordinare la lista, la scelta dipende dagli obiettivi dell'utente: se le informazioni sui prezzi delle immagini di destinazione sono state inserite correttamente, si potrebbe decidere di basarsi su tale parametro e prendere quindi l'immagine con il costo più basso. Questa è una scelta molto generica, ma, in casi particolari, si potrebbero anche effettuare ordinamenti più raffinati, combinando i parametri singoli disponibili per l'immagine di destinazione.

Infine, la quarta e ultima fase dell'algoritmo prevede, una volta ottenuto un Dictionary che associa a ogni immagine di input migrabile una e una sola immagine cloud, il suo ordinamento. In questo caso, la differenza rispetto all'ordinamento precedente sta nel fatto che l'ordinamento può essere fatto su parametri dell'immagine di origine, su quella di destinazione oppure, nello scenario più flessibile, su una combinazione di essi.

Al termine dell'esecuzione dell'algoritmo, si avrà quindi una lista ordinata dalla quale si potrà iniziare a procedere in ordine con la migrazione o che potrà essere ulteriormente raffinata, prendendo in considerazione i workload; opzionalmente, si può anche ottenere la lista complementare delle immagini, per cui la migrazione in cloud non è possibile in base ai parametri presi in considerazione.

4.5.1 Parametri

Come si è detto, nell'algoritmo sono stati volutamente lasciati dei gradi di libertà per l'utilizzatore, che può così influire sull'output prodotto; essi vengono spiegati di seguito.

Il primo grado di libertà è sul numero di parametri sui quali fare il matching: potenzialmente, infatti, si potrebbero analizzare tutti i parametri del modello dei dati che sono presenti sia sull'immagine in input che su quella di destinazione. La scelta di quali di essi utilizzare dipenderà sia dall'ampiezza dei dati in input sia dal livello di raffinamento desiderato; infatti, se abbiamo a disposizione molti parametri, possiamo ridurre il numero di immagini prodotte da questa fase, ma all'altro estremo rischiamo di non ottenere alcun match se abbiamo troppi parametri.

Il secondo grado di libertà consiste nell'ordinamento delle immagini compatibili, per ottenere il *best match*: come accennato poco fa, la scelta più semplice potrebbe essere quella del costo dell'immagine di destinazione, ma nulla vieta di utilizzare, se disponibili,

Algoritmo 2 Algoritmo che per ogni immagine in input ci restituisce il best match

```
Dictionary<Input_image, Cloud_image> results = new Dictionary();
foreach (input_image in input_images) {
  cloud_images_compatible = new ArrayList();
  os_target = OSMatch.get(input_image.OS);
  if (os_target is null) then break("Input image is not migrable due to OS"); //we will
      return a list containing skipped images

  foreach (cloud_image in cloud_images) {
    if (cloud_image.OS != os_target) then break("Cloud image not compatible due to
        OS");
    if (cloud_image.OS_bits != input_image.OS_bits) then break("Cloud image not
        compatible due to OS bits");
    if (cloud_image.num_cpu < input_image.num_cpu) then break("Cloud image not
        compatible due to CPU");
    if (cloud_image.RAM < input_image.RAM) then break("Cloud image not compatible
        due to RAM");
    if (cloud_image.storage_size < input_image.storage_size) then break("Cloud image
        not compatible due to Storage size");
    //CPU frequency...
    cloud_images_compatible += cloud_image;
  }
  cloud_images_compatible.sort(); //by price, location or something else
  best_match = cloud_images_compatible.first();
  results.put(input_image, best_match);
}
results.sort(); //now sort the matched input, e.g. by efforts needed
```

altri parametri collegati all'immagine in cloud o specifici dei singoli provider, come, per esempio, la locazione dei datacenter oppure il Service Level Agreement garantito.

Il terzo e ultimo grado di libertà è quello sull'ordinamento delle immagini: in questo caso è possibile effettuare l'ordinamento in base a una lista di parametri piuttosto ampia, provenienti sia dall'immagine originale che da quella scelta come miglior target. Se, ad esempio, si volesse puntare al minor costo delle immagini, si potrebbe ordinare ancora una volta in base a questo parametro, per cui la migrazione inizierebbe dai tagli di istanze più piccoli. Un'altra scelta che potrebbe soddisfare alcuni casi può essere quella di dare maggior priorità a tutte le immagini per le quali non c'è necessità di cambiare o aggiornare il sistema operativo oppure a quelle che contengono il minor numero di software.

4.5.2 Difficoltà di migrazione

Un parametro interessante, che è possibile associare ad ogni immagine segnalata come migrabile, è un coefficiente che riassuma lo sforzo richiesto per tale processo di migrazione; su tale coefficiente si possono fare dei calcoli molto complessi. Per esempio in [TLF+11] si propone una stima basata sulla dimensione complessiva del progetto, nella quale si prendono in considerazione quattro parametri:

1. sforzo per la reinstallazione e la configurazione nell'ambiente di destinazione

2. cambiamenti necessari al database, sia strutturali che nel tipo stesso di database (per esempio il passaggio a un database non relazionale)

3. cambiamenti al codice per adeguare alla struttura richiesta

4. cambiamenti alla struttura delle connessioni tra gli elementi del progetto, per esempio nel caso di migrazione parziale.

Nel nostro caso, si è optato per un coefficiente numerico che si basa su criteri di più alto livello, senza concentrarsi sui cambiamenti alle singole immagini. Tale coefficiente è dato da $C_{tot} = C_{so} + N_{software} + Virt$, dove C_{so} è calcolato in base al costo richiesto dall'upgrade del sistema operativo, che è massimo nel caso si debba aggiornare a una versione non direttamente successiva o sia necessario passare da un sistema operativo Unix a un altro. Troviamo poi il numero di software ospitati dall'immagine di origine (da cui si possono escludere eventualmente i software non rilevanti per l'analisi, quali per esempio utility di compressione) e se l'immagine è già virtualizzata o meno.

Possiamo fare un esempio di utilizzo di questa metrica ipotizzando di dover analizzare da un'immagine già virtualizzata, dotata di sistema operativo Microsoft Windows Server 2003 e 10 programmi; se il cloud di destinazione offre immagini con sistema operativo Microsoft Windows Server 2008, per le quali la migrazione, partendo dalla versione in oggetto, ha, secondo l'analista, un costo 1, possiamo calcolare il costo totale come $C_{tot} = 1 + 10 + 0 = 11$. Se emergesse la necessità di dare più peso a uno dei fattori, si può pensare di aggiungere un coefficiente ai singoli parametri in base alle esigenze.

4.5.3 Estensione per l'analisi dei workload

Nel caso l'utente desideri effettuare un'ulteriore scrematura secondo l'approccio ai workload visto in sezione 4.4.2.2, è possibile partire dall'algoritmo precedente ed estenderlo fino ad arrivare a un procedimento come quello visibile nell'algoritmo 3.

Come input si prendono i risultati ottenuti in precedenza, per cui le immagini in input sono suddivise in due liste, quelle migrabili e quelle che non lo sono; da queste ultime possiamo estrarre i loro workload. Come si è detto, le visioni possibili sono due: quella indipendente prevede che, nel caso un workload includa sia immagini migrabili che non migrabili, venga spezzato in due parti; quella combinata, invece, blocca la migrazione di un intero workload se almeno una delle immagini che lo compongono non è migrabile. Quest'ultimo scenario è quello preso in considerazione dall'algoritmo, che scarterà quindi anche tutte le immagini che, pur avendo caratteristiche compatibili con le immagini di destinazione, appartengono a workload segnati come non migrabili. Al termine otterremo dunque una lista delle immagini che appartengono a workload completamente migrabili, ordinate secondo i criteri scelti nell'algoritmo precedente.

Algoritmo 3 Algoritmo che elimina le immagini appartenenti a workload non migrabili

```
ArrayList<Input_Image> images_not_compatible; //images that cannot be migrated
ArrayList<Input_Image> images_compatible; //images that can be migrated
ArrayList<Workload> workloads_not_compatible;
foreach (image in images_not_compatible) {
  //extract workloads
  workloads_not_compatible += (image.workload);
}
foreach(image_ok in images_compatible) {
  //remove each image for which workload is not migratable
  if (image_ok.workload is in workloads_not_compatible) then images_compatible.
      remove(image_ok);
}
return images_compatible;
```

4.6 Fasi di una migrazione

Come rappresentato in figura 4.5, la migrazione può essere organizzata in sei fasi, suddivisibili in due parti autonome: la prima fase è quella del discovery automatico dell'ambiente

originario (e eventualmente dei provider di cloud e delle offerte che essi forniscono, se essi offrono un'interfaccia apposita o dei metodi di esplorazione). C'è poi l'analisi dei dati estratti, per ripulirli e ordinarli in base alle esigenze dei singoli casi. La terza fase è quella del matching, che può avvenire su tre livelli: quello delle relazioni tra workload e immagini, quello delle caratteristiche delle singole immagini e infine il livello dei software. A questo punto viene prodotto un risultato con le migliori offerte corrispondenti alle richieste e lo sforzo richiesto per la migrazione.

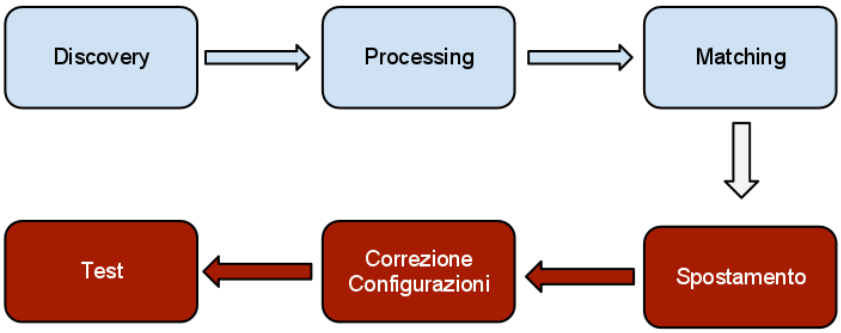

Figura 4.5: Le fasi di una migrazione

A questo punto si apre la seconda parte, quella della migrazione vera e propria, con le diverse fasi di spostamento delle immagini sul cloud, correzione delle configurazioni esistenti e infine l'ultima che corrisponde ai test delle prestazioni del sistema migrato con eventuali aggiustamenti, se necessario. Quest'ultima fase è quella più dinamica e verrà in parte discussa nel capitolo successivo.

4.6.1 Discovery

La fase di discovery è la prima del processo di migrazione; in generale, il discovery di una rete di sistemi può essere effettuato in maniera automatica o in maniera manuale: la maniera manuale prevede l'inserimento dei dati nello strumento previsto per l'input e risulta fattibile per un numero di immagini limitato, mentre strumenti automatici offrono una scalabilità molto più elevata e sono quelli più diffusi in ambito enterprise. Una rapida panoramica degli strumenti esistenti a livello enterprise con anche le suddivisioni esistenti è stata fatta in sezione 3.2.1.2.

Nel nostro progetto si utilizzano come input dei fogli di calcolo in formato Microsoft Excel, la cui popolazione può avvenire a partire da query su di una soluzione IBM Tivoli CCMDB o equivalente. È importante, tuttavia, separare la parte di parsing dall'analisi vera e propria, per non escludere in futuro soluzioni più dirette come il collegamento diretto a un database o a strumenti diversamente impostati.

Nel fare l'input di fogli di calcolo Excel è possibile seguire due strade: la prima è quella di avere un file di esempio con una determinata struttura, la seconda quella di permettere un input libero, ma andare a modificare il motore di parsing in base a esso. La prima scelta offre una serie di vantaggi, fra i quali maggior facilità d'uso e applicabilità dello strumento, tuttavia per motivi di tempo non è stato impostato tale file, ma la soluzione è stato impostata in modo da poterlo fare in futuro senza necessità di grossi cambiamenti.

4.6.2 Processing

Questa fase include tutte le operazioni che è necessario effettuare sui dati prima di passare all'analisi vera e propria; la sua necessità varia in base al caso che si sta affrontando: potrebbero, per esempio, essere richieste operazioni di pulizia sui dati come l'eliminazione di immagini in input non valide, la normalizzazione di parametri o altri passi simili.

Il processing è potenzialmente effettuabile in due momenti: a livello di database, quindi a monte del nostro strumento o al momento del parsing dei dati e suo inserimento nel modello dei dati previsto. Nel nostro strumento si è scelto di integrare una parte dimostrativa che si occupa di questo compito, per cui già al momento della lettura dei file in input vengono scartati degli elementi privi delle caratteristiche minime; per fare un esempio, immagini prive di sistema operativo non permettono un matching adeguato e quindi non vengono prese in considerazione nelle fasi successive. Vengono inoltre previsti dei controlli sulla correttezza dei valori, come per esempio il numero di bit del sistema operativo. Al termine di questa fase possiamo dunque essere sicuri di avere dei dati affidabili e sui quali potremo procedere con le elaborazioni vere e proprie.

4.6.3 Matching

In questa fase, che è quella più importante, verranno presi in considerazione tutti i parametri descritti nella sezione precedente, cui assegniamo dei coefficienti in modo da dare più

importanza a una o all'altra caratteristica. L'idea alla base è, quindi, di fare un confronto per ogni immagine con quelle presenti sul cloud di destinazione, in modo da ottenere quelle più economiche che corrispondano alle immagini di partenza. L'utente, se lo desidera, può anche specificare dei parametri non legati alla singola immagine, quali la locazione desiderata. Questo permette di ottenere, quindi, una lista delle immagini per le quali si può fare la migrazione, escludendo, per esempio, quelle per le quali il sistema operativo non è disponibile sul cloud, né vi sono versioni compatibili.

Dopo questa fase è possibile pensare all'ordinamento delle immagini ottenute in base ai criteri scelti, in modo da poter capire da quali immagini è possibile partire. Da questo punto di vista non è possibile dare indicazioni generali, perché ogni analista potrebbe valutare in modo diverso l'importanza dei singoli parametri. Nei nostri test, per esempio, si è scelto di produrre nei risultati la difficoltà richiesta per la migrazione a un nuovo sistema operativo e il numero di software presenti su ogni immagine, ma ovviamente è possibile combinare a piacere i parametri presi in considerazione all'interno del nostro modello dei dati.

4.6.4 Spostamento

Una volta scelte le immagini che hanno più probabilità di essere migrate, inizia la seconda parte dell'operazione di migrazione, quella della migrazione effettiva. In questo caso gli approcci possibili sono suddivisibili in due categorie: la prima è quella della migrazione "rapida", che consiste nel trasporto dell'immagine virtuale dall'ambiente di origine a quello cloud, la seconda è quella della migrazione per reinstallazione.

Il primo di questi due scenari è appunto quello più veloce, ma in base ai singoli provider ha una serie di prerequisiti che non lo rendono una soluzione universale, in particolare:

- se il sistema di origine è fisico, esso va prima virtualizzato con strumenti specifici quali VMware vConvert

- il cloud di destinazione deve permettere l'importazione di immagini virtuali e supportare il sistema operativo installato sulla macchina; nel caso dell'SCE, per esempio, l'operazione è possibile solo per sistemi operativi Red Hat Enterprise Server e Suse Enterprise Linux, mentre altri fornitori offrono maggior flessibilità

- tutti i requisiti specifici sulle configurazioni macchine da importare devono essere soddisfatti; sempre restando all'interno dell'offerta IBM, per esempio, è necessario un kernel sufficientemente moderno con supporto *virtio* attivato a livello del kernel
- è necessario verificare che le licenze dei software installati consentano tale passaggio.

Una volta verificato il rispetto di tali requisiti, si può passare all'importazione vera e propria che, a seconda dei casi, può richiedere la preparazione dei metadati necessari prima dell'esecuzione dei comandi o può invece offrire strumenti grafici in grado di semplificare il processo.

La soluzione della reinstallazione, invece, è applicabile in ogni situazione, anche se ovviamente può richiedere un tempo molto maggiore, in base al numero di software installati e alle possibilità di esportazione e reimportazione che essi offrono. In questa situazione è possibile dividere i compiti da eseguire in due parti: la prima comprende l'istanziazione di una macchina con capacità equivalente a quella originale e la reinstallazione dei vari software, la seconda l'importazione dei due elementi fondamentali: le configurazioni e i dati.

Per quanto riguarda la reinstallazione dei software, per evitare la tediosa operazione di reinstallazione di ogni singolo componente, un approccio possibile potrebbe essere quello della ricerca nel cloud di destinazione di immagini che li contengono. Infatti tutti i principali provider IaaS offrono un catalogo che, oltre a semplici immagini configurate con il solo sistema operativo, contiene anche delle immagini con preinstallato del software. Dal momento che è impossibile soddisfare tutte le possibili combinazioni di software, alcuni produttori si stanno orientando verso strumenti che permettono di creare dei *bundle*, cioè di comporre a piacimento dei software di base, in modo da ottenere la configurazione completa desiderata. Si potrebbe dunque pensare di fare una ricerca nel catalogo delle immagini o preparare dei bundle con strumenti di questo genere per risparmiare del tempo, specialmente se la configurazione da migrare è piuttosto comune, come per esempio un application server o un database.

Per la seconda parte, quella specifica di ogni singola installazione e che dal punto di vista del business ha maggior valore, la migrazione dei dati in genere è ben coperta dagli strumenti propri dei vari software; per quanto riguarda la migrazione delle configurazioni, invece, per evitare un intervento completamente manuale, lungo e con possibilità di erro-

re, è necessario ricorrere a utility quali quelle presentate in sezione 3.2.1.4, in grado di esplorare le configurazioni di partenza e copiarle nel sistema di destinazione. Strumenti di questo tipo sono distribuiti commercialmente sia a livello consumer che enterprise e il loro limite consiste principalmente nella necessità di supportare manualmente i programmi desiderati, per cui si dovrà creare un database delle regole.

4.6.5 Correzione delle configurazioni e dei permessi

Una volta reinstallati i software e migrate le configurazioni, un'ulteriore problematica è quella della necessità di aggiornare le configurazioni al nuovo ambiente di rete. Questa fase potrebbe eventualmente essere anche fusa con quella precedente, nella quale vengono migrate le configurazioni, ma i due elementi sono concettualmente separati. Per questo compito è stata pensata una serie di strumenti, che abbiamo brevemente analizzato in 3.2.1.4.

L'idea alla base è quella di cercare di approntare un'euristica in grado di rilevare tutti i possibili parametri dinamici e poi sostituirli con i nuovi valori. Per fare un esempio piuttosto comune, molte applicazioni potrebbero contenere nelle loro configurazioni degli indirizzi IP, ad esempio quello di un database al quale accedono; dato che al momento gli ambienti cloud offrono la possibilità di avere sì un indirizzo IP statico ma tale indirizzo non è comunque personalizzabile dall'utente, ma scelto all'interno del pool di indirizzi a disposizione del provider, sarà necessario modificare tutti gli indirizzi IP esistenti con quelli nuovi assegnati dal provider cloud. Un problema simile potrebbe essere quello dell'hostname di una specifica macchina, visto che alcune applicazioni effettuano dei controlli sulla licenza proprio in base a questo parametro.

4.6.6 Test

La fase finale, prima dell'entrata in produzione del sistema migrato, è quella di verifica del corretto funzionamento e delle prestazioni. Il corretto funzionamento è ovviamente molto importante, soprattutto dal lato della configurazione dei middleware presenti: un'opzione non aggiornata o che ha un effetto diverso in una nuova versione del middleware stesso, potrebbe avere conseguenze imprevedibili ed essere difficile da rilevare, in particolare se

il problema si riscontra solo in casi limitati; strumenti automatici, come quelli visti nella sezione precedente, possono evitare molti di questi casi.

Il test delle prestazioni è invece un'esigenza particolarmente sentita in ambito cloud, dove le prestazioni non sono garantite. In caso le prestazioni non siano sufficienti, sarà necessario effettuare la migrazione a un'istanza più performante. Va notato come questa operazione potrebbe non essere banale in un ambito di private cloud, dove è possibile avere una scelta delle istanze maggiormente raffinata, per cui è possibile aumentare in maniera granulare sia la quantità di RAM sia il numero di CPU a disposizione. In questo caso trovare il giusto compromesso potrebbe essere un'operazione abbastanza lunga, in quanto sarebbe necessario provare tutte le varie combinazioni, per cui è da valutare se sia più conveniente trovare l'istanza che rispetta precisamente i requisiti oppure accontentarsi di un'istanza che sia in grado di soddisfare tali requisiti senza essere quella con il minor costo possibile.

4.7 Altri aspetti di interesse

4.7.1 Differenze nella migrazione verso un cloud pubblico o privato

Anche se offrono livelli di sicurezza e dall'altro lato costi molto diversi, da un punto di vista statico questi due modelli di deployment sono abbastanza simili. Nel caso del cloud pubblico, la problematica principale sta nelle caratteristiche fisse delle immagini disponibili sul cloud; occorrerà quindi fare delle considerazioni su tali dimensioni, sui sistemi operativi che esse dispongono e cercare di adattare nel miglior modo possibile, rispettando criteri che in genere saranno basati sul costo. Va fatta anche attenzione nel passaggio da CPU fisiche dei server *in house* a quelle virtuali offerte dalle soluzioni cloud, per le quali vengono dichiarate in genere delle frequenze da verificare in fase di test.

Nel caso di un cloud privato, invece, siamo costretti in maniera inferiore a sottostare a limiti imposti sulle caratteristiche delle immagini, anche se possono rimanere quelli legati ai sistemi operativi supportati e alle applicazioni in base al software scelto per la gestione dell'ambiente di virtualizzazione, per esempio vSphere di VMware o TSAM in casa IBM; è quindi possibile avere una maggiore granularità di scelta, anche se permangono i suggerimenti di verificare le prestazioni in ogni caso.

Da un punto di vista generale, inoltre, una migrazione a un cloud privato, di solito, prevede un numero decisamente maggiore di immagini in input da analizzare. Questo sia per la struttura tipica della struttura enterprise, che prevede in genere una topologia complessa, sia per la stessa convenienza della creazione di un cloud privato, che non avrebbe senso se le istanze da portare in cloud fossero poche decine, poiché non si riuscirebbe a sfruttare l'economia di scala e a ripagare gli investimenti iniziali sull'infrastruttura. Ciò ha influenza più che altro sulla raccolta dei dati, che è impensabile fare in maniera manuale e sulla chiarezza dell'output: visti i numeri elevati, devono essere chiaramente individuabili i motivi per cui alcune immagini sono state eliminate nelle fasi di processing oppure non sono state inserite nella lista in output.

Un ultimo fattore di variabilità è quello legato all'importanza da assegnare ai diversi parametri da prendere in considerazione: ciò verrà tenuto in considerazione permettendo la scelta di coefficienti da assegnare ai diversi fattori, il che ci permette di mantenere in ogni caso un framework condiviso che sia valido sia per cloud pubblici che privati.

4.7.2 Indipendenza dal cloud provider

Nel corso del nostro lavoro abbiamo cercato di mantenere sempre una visione generalistica e non dipendente dalle caratteristiche dei singoli provider, anche se naturalmente la maggior parte del lavoro e dei test sono stati effettuati sulla soluzione IBM SCE; questo requisito non funzionale è stato affrontato da due punti di vista, spiegati brevemente di seguito.

Nell'ambito dell'analisi ciò implica che, per lo sviluppo del nostro modello dei dati, sono state analizzate le offerte di più provider di cloud e si è cercato di individuare delle caratteristiche comuni a tali offerte, evitando, per quanto possibile, quelle specifiche; sono inoltre stati aggiunti alcuni parametri che possono essere specifici per i singoli provider, quali la locazione dei datacenter.

Dal punto di vista della migrazione vera e propria - la seconda delle due parti individuate poco sopra quindi - la scelta fatta è di analizzare e sfruttare, per quanto possibile, delle API in grado di astrarre le differenze tra i vari provider, in modo da facilitare l'applicazione delle nostre idee ad ambiti differenti. Dall'altro lato, tali API offrono il massimo comun denominatore tra le diverse funzionalità dei provider, per cui questo limita le possi-

bilità di azione alle caratteristiche di base, impedendo eventualmente di sfruttare funzioni avanzate, che sono invece offerte dalle singole API proprietarie.

4.8 Analisi dinamica

Oltre all'analisi appena presentata, che si configura come prettamente statica, è possibile prevedere una fase più dinamica, nella quale il criterio di giudizio siano le prestazioni di applicazioni reali destinate alla migrazione. Questo procedimento può essere associato a quella statica in più modi: nel caso di un campione di poche immagini, potremmo pensare di evitare completamente l'analisi precedente per affrontare direttamente un test prestazionale. Nel caso di un numero maggiore di immagini da migrare, invece, è possibile introdurre un processo in pipeline: per prima cosa, effettuo un'analisi statica con CloMAs o uno strumento equivalente, che mi permette di scartare le immagini sicuramente incompatibili con l'ambiente di destinazione e, in base ai parametri che l'analista ha fornito, produce una lista ordinata di immagini da migrare. A partire da queste indicazioni, posso in seguito passare alla fase di test, che inizierà, quindi, dalle immagini che, in base ai criteri scelti, sono più convenienti per la migrazione.

4.8.1 Passi operativi del processo

Come detto, questa fase, rispetto a quella di analisi statica, è molto meno standard e possiamo fare solo delle considerazioni molto generali sui passi previsti, che riassumiamo in figura 4.6. In particolare, quello che dipenderà dal singolo caso sarà il criterio per determinare se le prestazioni di un sistema sono adeguate alle esigenze.

In genere si può pensare di partire o dalla macchina virtuale suggerita dallo strumento statico oppure considerare solo gli elementi distintivi di base (sistema operativo e numero di bit) e quindi scegliere l'immagine più piccola disponibile sul cloud di destinazione.

Si fa quindi lo spostamento vero e proprio in ambiente cloud che, come si è detto in precedenza, può avvenire o in maniera rapida importando le immagini virtuali, con i vincoli esaminati oppure per reinstallazione. Una volta effettuato lo spostamento, è necessario correggere eventualmente le configurazioni, per esempio indirizzi IP che puntano ad altre macchine oppure le regole dei firewall. Al termine, quando si ha un ambiente funzionante, si può procedere con l'analisi delle prestazioni.

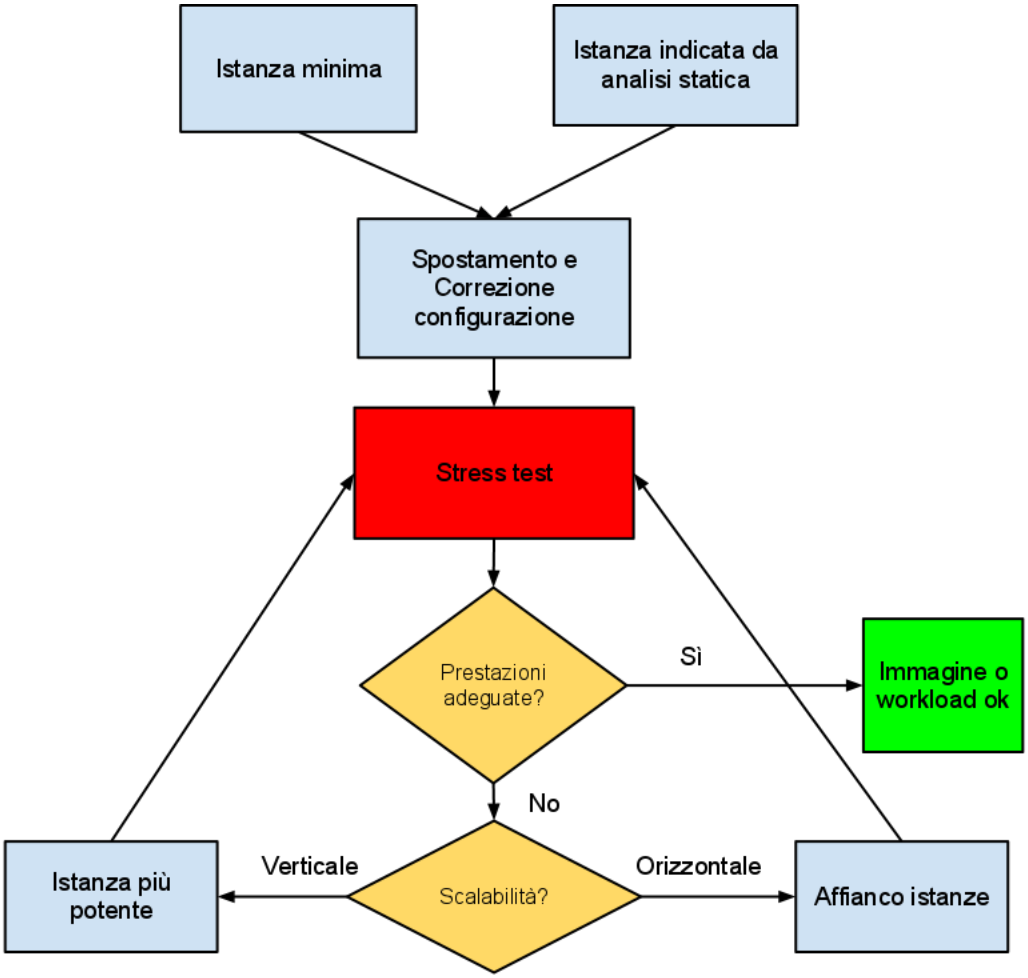

Figura 4.6: I passi previsti nell'analisi dinamica

4.8.1.1 Stress test

La fase di test è chiaramente molto importante, in quanto deve essere in grado di simulare, nella maniera più completa, quello che potrà essere il carico a cui è sottoposta la macchina. Se, per esempio, il nostro interesse è quello di sfruttare la capacità computazionale dell'istanza creata, per esempio per applicazioni matematiche, potremmo pensare a un benchmark classico in grado di valutare il numero di operazioni eseguibili in un periodo di tempo. Nel caso invece di un'applicazione Web, ha più senso capire il numero di richieste gestibili dal sistema. Va fatto notare come si possa dimensionare in base alle necessità normali, a differenza di macchine tradizionali dove era necessario dimensionare

per i picchi, con conseguente sottoutilizzo delle macchine.

Un ulteriore passo potrebbe essere quello di valutare queste prestazioni in termini di costi; infatti, per la struttura stessa di un sistema cloud, i costi orari sono calcolabili in maniera abbastanza semplice.

Questa fase, in base ai risultati, potrebbe richiedere più interazioni, mentre, nel caso migliore, la prima immagine sottoposta a prove garantirà già il raggiungimento delle prestazioni desiderate.

4.8.1.2 Scalabilità

In caso di prestazioni non adeguate ai requisiti, si cerca una configurazione più potente e ciò può essere fatto scalando le risorse in due modi: verticale o orizzontale.

La prima modalità prevede, in caso di prestazioni non sufficienti, di passare a un'istanza più potente, quindi con maggior numero di CPU e/o RAM. Questo approccio è quello classico, in quanto tradizionalmente la prima misura era quella di estendere le caratteristiche di un server. Il limite maggiore è che c'è un limite a tale espansione, visto che, anche se le capacità computazionali aumentano continuamente, potrebbero non essere sufficienti per una crescita particolarmente intensa; tale limite è anche più evidente in ambiente cloud, dove le dimensioni delle istanze possono essere scelte solo all'interno di una rosa limitata.

La seconda modalità, invece, prevede, in caso di limiti prestazionali, di aggiungere delle istanze in parallelo. Tale possibilità è in genere più complessa dal punto di vista architetturale, ma, se si riesce a parallelizzare tutti gli elementi del sistema, consente una scalabilità quasi infinita. L'ulteriore vantaggio è che tale scalabilità è dinamica, in quanto è molto facile mettere in piedi un meccanismo di load balancing dinamico in grado in breve tempo di aggiungere o rimuovere le istanze in base a parametri scelti dall'utente; tale meccanismo è, per esempio, fornito come servizio già pronto, di nome Elastic Load Balancing[1], da Amazon, mentre su SCE è possibile creare un'architettura del genere sfruttando le capacità di IBM Tivoli Monitoring. Un cambiamento di questo genere è invece più complicato, soprattutto nel caso di riduzione della capacità richiesta, in uno scenario di scalabilità verticale.

[1] http://aws.amazon.com/elasticloadbalancing/

Capitolo 5

Dettagli implementativi e test

Una volta descritta la parte di progetto e le idee alla base degli strumenti sviluppati, proseguiamo in questo capitolo su due fronti principali:

- analizzeremo le scelte implementative fatte per quanto riguarda lo strumento di analisi statica
- approfondiremo nello scenario di analisi dinamica il tema della scalabilità, che ha un notevole impatto al momento della migrazione verso il cloud.

Per entrambe le parti, passeremo poi alla fase di verifica dei risultati raggiunti, che verranno valutati in base a criteri di correttezza e raggiungimento dei criteri prestazionali desiderati; tali criteri andranno eventualmente calibrati in base ai singoli casi affrontati.

5.1 Backend modello dei dati

Il primo elemento da sviluppare per l'analisi statica è il modello dei dati, che è al centro dello strumento di migrazione. Il modello dei dati astratto, presentato in sezione 4.3, è stato implementato nella pratica tramite un database relazionale. Questa scelta, che si discosta da quella di altri strumenti analizzati, è stata fatta sia in base alla visione di un modello relazionale come naturale implementazione del nostro modello dei dati, sia allo scopo di permettere una maggior flessibilità all'utilizzatore; egli potrà interagire, tramite delle query, con il modello dei dati in maniera personalizzata, evitando un approccio a *black box,* che è troppo rigido in un ambiente caratterizzato dall'estrema variabilità sia per

quanto riguarda i dati di origine che in quelli richiesti in output. La scelta di un modello relazionale vuole anche sottolineare come i dati siano al centro del nostro progetto e ne siano la parte più importante, mentre l'input e l'output devono essere visti come componenti intercambiabili.

Nella scelta del prodotto usato nel corso dei test si è cercato di mantenere aperte più possibilità, per cui sono state previsti due database, tra cui è possibile scegliere al momento dell'esecuzione, grazie all'astrazione che ci permette la tecnologia Java Database Connection (JDBC).

Il primo database preso in considerazione è HSQLDB[1]; questa è una soluzione molto portabile, distribuita come un singolo file *all inclusive*, che può essere dunque integrato all'interno del nostro strumento, per rendere possibile un processo di analisi anche in assenza di una connessione Internet. Esso supporta gran parte dello standard SQL-92 e quindi è una buona scelta anche da questo punto di vista, evitando, per quanto possibile, problemi di lock-in in previsione di una migrazione verso soluzioni più avanzate dal punto di vista tecnico e prestazionale.

Il secondo database messo a disposizione è DB2, un prodotto IBM che appartiene a una fascia completamente diversa, anche dal punto di vista puramente economico, in quanto pensato principalmente per un target di tipo enterprise. Dal punto di vista delle caratteristiche è decisamente più completo, offrendo funzionalità avanzate per la sicurezza e l'availability, oltre a prestazioni superiori in presenza di grandi moli di dati, al prezzo di una maggior complessità e una minore portabilità.

La scelta tra i due strumenti, come si è detto, può essere fatta al momento dell'esecuzione per mezzo di un parametro: a nostro parere, visto anche che la mole dei dati prevista in input non sarà mai eccessivamente grande (generalmente nell'ordine di qualche migliaia di righe), HSQLDB può essere la scelta migliore per analisi rapide da effettuarsi presso il cliente, magari in assenza di una connessione Internet affidabile, mentre, nell'ottica di un confronto di diversi casi e dunque della memorizzazione di tutti i dati analizzati in precedenza, ha senso sfruttare le caratteristiche aggiuntive offerte da DB2 o da prodotti simili e approntare quindi un database centralizzato, che potrebbe poi essere ospitato sia su un server tradizionale sia in un ambiente cloud. Quest'ultima soluzione è proprio quella adottata nel corso dei nostri test, in quanto ci ha permesso di ridurre al minimo i tempi di installazione e di manutenzione del prodotto, che era in ogni caso un aspetto secondario del

[1] HyperSQL DataBase http://hsqldb.org/

nostro lavoro; a partire dall'immagine presente nel catalogo di IBM Smart Cloud Enterprise, è stato infatti possibile creare un'istanza di DB2 in pochi minuti, con costi calcolati in base al reale utilizzo della macchina.

Per concludere, rimarchiamo come la buona aderenza di entrambi gli strumenti utilizzati abbia permesso di utilizzare le stesse query in entrambi i casi, per cui non è stato necessario scrivere codice specifico. L'utilizzo di funzioni peculiari di un prodotto oppure l'utilizzo di database meno aderenti agli standard potrebbero invece richiedere modifiche maggiori e quindi rendere più costoso un supporto contemporaneo.

5.2 Strumenti utilizzati

Per la nostra implementazione si è utilizzato il linguaggio Java nella sua versione 6, che gode di ampia diffusione a livello enterprise. Sono state utilizzate alcune librerie aggiuntive per interfacciarsi con l'esterno, in particolare Apache POI[2], che ci permette la lettura e la scrittura su fogli di calcolo in formato Microsoft Excel. Per quanto riguarda il modello dei dati, sono stati utilizzati i driver JDBC per il collegamento con l'istanza che ospita un database IBM DB2, mentre, per quanto riguarda HSQLDB, viene incluso nel progetto il package che ne permette anche l'uso embedded.

5.3 Architettura del progetto

Il codice è stato suddiviso in package, visibili in figura 5.1, in modo da permettere anche il riuso dei componenti al di fuori dei confini del singolo strumento sviluppato in questa occasione; tale requisito era piuttosto importante, soprattutto dal punto di vista dell'inserimento dei dati, una fase che anche durante i nostri test si è presentata in maniera molto eterogenea.

Grazie alla suddivisione prevista, sarà quindi possibile utilizzare gli algoritmi e gli strumenti di matching senza alcuna dipendenza con la fase di input e di parsing dei dati, aprendo la possibilità a metodi di popolamento alternativi. Di seguito i package previsti:

db include tutti i diversi componenti che si interfacciano al database, sia per il suo popolamento che per quanto riguarda le interrogazione predefinite o personalizzate

[2] http://poi.apache.org/

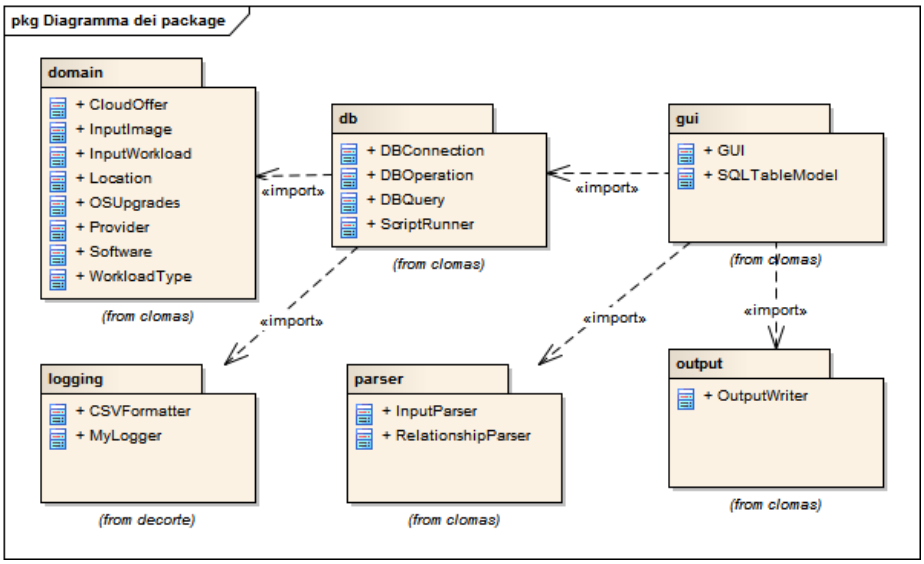

Figura 5.1: Diagramma dei package di CloMAs

domain contiene le classi corrispondenti al modello dei dati descritto in dettaglio in sezione 5.1. Nel nostro progetto, dato che la struttura realizzata è relativamente semplice, non si è fatto uso di strumenti di tipo ORM

gui comprende tutti i componenti relativi all'interfaccia grafica, implementata con il framework Swing e che utilizza per l'elaborazione vera e propria i metodi esposti dalle altre classi

output in questo package vengono raccolti i metodi per produrre degli output su disco; nel nostro caso è stata implementat una classe in grado di esportare i risultati di un'interrogazione in un foglio di calcolo in formato Microsoft Excel, così da permettere ulteriori elaborazioni

parser qui vengono raccolti tutti gli strumenti per la lettura dei dati (ed eventualmente operazioni di pulizia e riconciliazione), che precedono la fase di inserimento nel database; nel nostro caso viene incluso anche uno strumento per il parsing delle relazioni, che implementa l'algoritmo 1

logging package che include strumenti destinati alla produzione di un log dell'attività, utile per la comprensione di eventuali errori e per la raccolta di dati sulle prestazioni.

5.4 Interfaccia utente

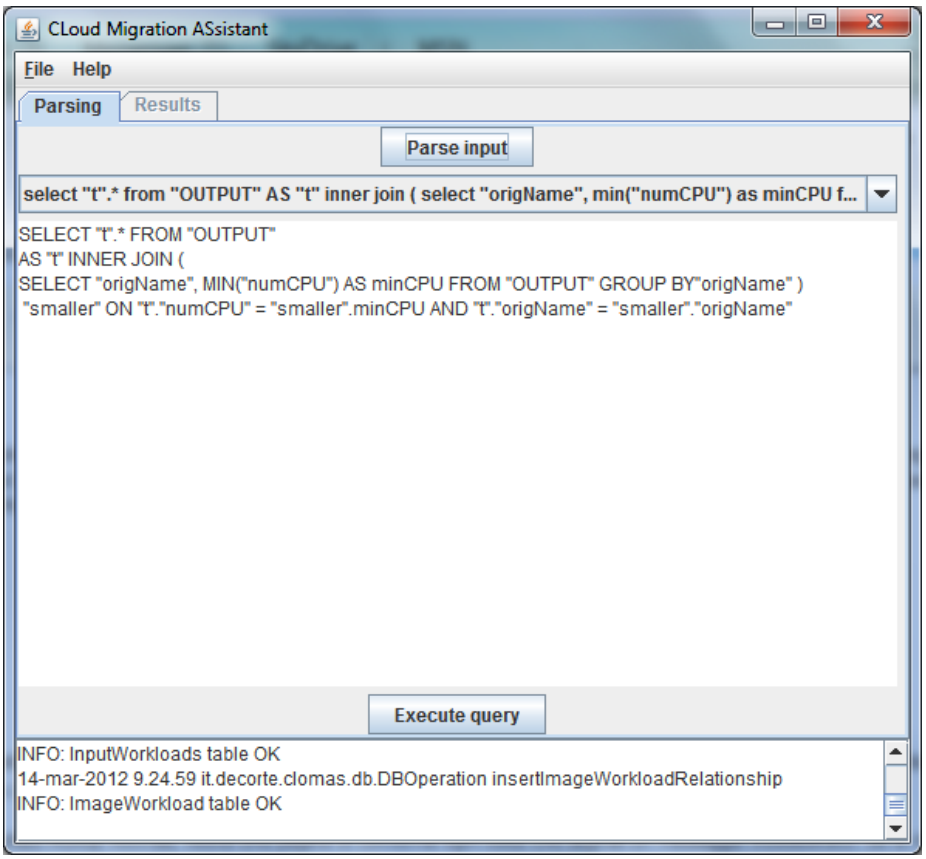

Figura 5.2: La prima scheda dell'interfaccia utente di CloMAs

L'interfaccia studiata per il nostro strumento è molto semplice ed può essere suddivisa in due sezioni fondamentali: la prima comprende le fasi di parsing e di esecuzioni di query, la seconda prevede la visualizzazione e l'eventuale esportazione dei risultati ottenuti. Nella prima sezione, visibile in figura 5.2, vediamo in alto un pulsante dedicato all'avvio della fase di parsing e al caricamento nella base di dati che, come già menzionata, sarà realizzata con il prodotto scelto al momento dell'avvio.

Tale scelta, ovviamente, non offre alcuna possibilità di configurazione, in quanto i file da leggere devono avere dei nomi prefissati. Nel caso dell'input, infatti, si hanno due possibilità: avere un formato comune per i dati di ingresso, eventualmente specificando anche le colonne previste e la loro posizione, oppure lasciare massima libertà sul formato, al prezzo di costringere l'utente a intervenire ogni volta sul codice di parsing. Visto il

target dello strumento, si è optato per la prima scelta, lasciando come sviluppo futuro la configurabilità dei nomi dei file da prendere in input.

Il secondo compito dell'interfaccia è l'inserimento di una query sul modello dei dati, eventualmente selezionabile tra quelle approntate in precedenza per mezzo di un apposito menu a discesa. Tale interrogazione viene poi eseguita sul database di riferimento.

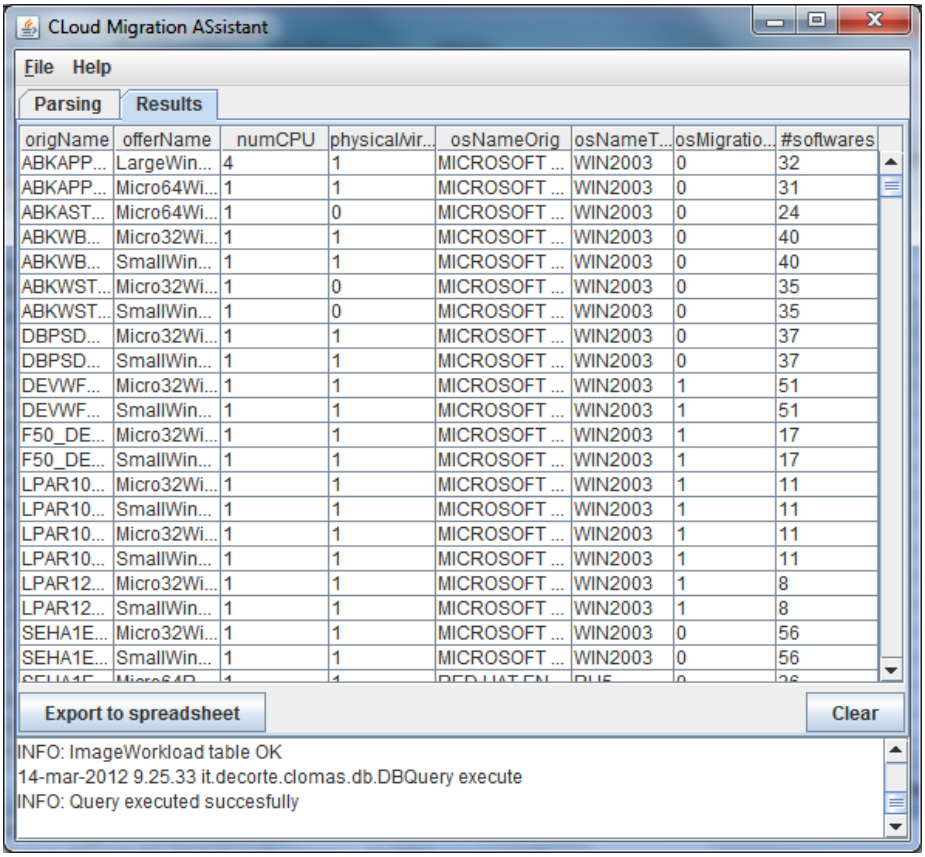

Figura 5.3: La seconda scheda con la visualizzazione dei risultati

Infine c'è la seconda sezione, visibile in figura 5.3, dove vengono presentati i risultati prodotti dall'interrogazione, con possibilità di esportarli tramite un apposito pulsante in un foglio di calcolo Excel.

In entrambe le sezioni possiamo notare come, in basso, sia sempre visibile uno spazio destinato alla visualizzazione dei log dell'attività dello strumento, che risulta anche molto importante per la comprensione di eventuali errori riscontrati nelle diverse fasi di analisi.

5.5 Interrogazioni

Come detto, nel progetto di CloMAs si è deciso di lasciare la massima libertà all'utente, per cui sul modello dei dati è possibile eseguire qualsiasi query SQL valida, sfruttando se necessario anche costrutti avanzati. In questo modo è possibile implementare algoritmi particolari, differenti da quello presentato in sezione 4.5, ottenendo in pratica un disaccoppiamento tra il motore di esecuzione e l'input che gli viene fornito, che è il nucleo principale del nostro sistema.

Di seguito, presentiamo come esempi alcune query che sono state ideate e utilizzate nel corso dei test e che rappresentano delle realizzazioni degli algoritmi di matching discussi in dettaglio nel capitolo precedente.

5.5.1 Immagini con complessità

Come primo esempio di possibile output presentiamo una lista che associ, a ogni immagine in input che sia compatibile con il cloud target, il suo best match, sulla base di un criterio che cerca di minimizzare il numero di CPU.

Il primo passo consiste nella creazione di una vista, a cui assegniamo il nome `output` e definita come segue. Le informazioni che verranno fornite dalla vista possono ovviamente essere scelte a piacere, ma nel nostro caso si è scelto di fornire il nome dell'immagine originale, quello dell'offerta cloud corrispondente, il numero di CPU dell'immagine originaria, se essa era fisica o già virtualizzata; ci sono poi i sistemi operativi di origine e destinazione, con lo sforzo di migrazione ad esso associato. Infine c'è il numero di software presenti sull'immagine di origine.

```
CREATE VIEW output ("origName", "offerName", "numCPU", "physical/virtual", "
    osNameOrig", "osNameTarget", "osMigrationEffort", "#softwares")
AS (
SELECT DISTINCT "InputImage"."imageName", "Offer"."imageName", "Offer"."numCPU
    ", "InputImage"."physical", "InputImage"."osName", "Offer"."osName", "OSUpgrades"
    ."effort",
(SELECT COUNT(*) AS softwareNumber FROM "InputSoftware" WHERE "InputImage"."
    id" = "InputSoftware"."idImage")
FROM "Offer", "InputImage", "OSUpgrades"
```

```
WHERE "InputImage"."osBits" = "Offer"."osBits" AND ("InputImage"."osName" = "
    OSUpgrades"."originalOSName" AND "OSUpgrades"."destinationOS" = "Offer"."
    osName") AND "InputImage"."numCPU" <= "Offer"."numCPU");
```

Tale vista associa a ogni immagine di partenza una o più immagini tra quelle offerte in cloud prese in considerazione; per questo motivo, è necessario applicare un'ulteriore interrogazione che, tra le immagini di destinazione compatibili, selezioni quella con il numero minimo di CPU.

```
SELECT "t".* FROM output AS "t"
INNER JOIN ( SELECT "origName", MIN("numCPU") AS minCPU FROM "OUTPUT"
    GROUP BY "origName" ) "smaller"
ON "t"."numCPU" = "smaller".minCPU AND "t"."origName" = "smaller"."origName"
```

5.5.2 Immagini che non richiedono l'aggiornamento del sistema operativo

Sulla stessa falsariga della query precedente, è possibile interrogare la base di dati in modo da ottenere solo le immagini per cui non è necessario effettuare l'aggiornamento a una versione più recente del sistema operativo; prendiamo, inoltre, come parametro per la selezione del best match il prezzo minimo, che richiede solo piccole modifiche rispetto al caso precedente, dove consideravamo invece il numero di CPU. La scelta di questo specifico parametro dipenderà dalla completezza dei dati raccolti, poiché, ad esempio, i costi potrebbero non essere disponibili.

```
CREATE VIEW output ("origName", "offerName", "cost", "physical/virtual", "osNameOrig
    ", "osNameTarget", "osMigrationEffort", "#softwares")
AS (
SELECT DISTINCT "InputImage"."imageName", "Offer"."imageName", "Offer"."cost", "
    InputImage"."physical", "InputImage"."osName", "Offer"."osName", "OSUpgrades"."
    effort",
(SELECT COUNT(*) AS softwareNumber FROM "InputSoftware" WHERE "InputImage"."
    id" = "InputSoftware"."idImage")
FROM "Offer", "InputImage", "OSUpgrades"
WHERE "InputImage"."osBits" = "Offer"."osBits" AND ("InputImage"."osName" = "
    OSUpgrades"."originalOSName" AND "OSUpgrades"."destinationOS" = "Offer"."
```

```
osName") AND "InputImage"."numCPU" <= "Offer"."numCPU" AND "InputImage"."
    RAM" <= "Offer"."RAM");
```

La query di selezione diventa la seguente.

```
SELECT "t".* FROM "OUTPUT"
AS "t" INNER JOIN ( SELECT "origName", MIN("cost") AS minCost FROM "OUTPUT"
    GROUP BY "origName" ) "smaller" ON "t"."cost" = "smaller".minCost AND "t"."
    origName" = "smaller"."origName" WHERE "osMigrationEffort" = 0
```

5.5.3 Immagini appartenenti a workload completamente migrabili

Come descritto in dettaglio in sezione 4.4.2.2, è possibile avere una visione dei workload combinata oppure indipendente. Nel primo caso, se anche una sola delle immagini di un workload non può essere migrata, anche tutte le altre appartenenti allo stesso workload dovranno essere escluse, con conseguenze a catena sugli altri workload collegati; ciò è stato definito in maniera teorica nell'algoritmo 3, del quale presentiamo l'implementazione pratica.

Creiamo inizialmente una lista complementare a quella delle immagini migrabili e da essa estraiamo gli id univoci dei workload cui tali immagini appartengono.

```
CREATE VIEW "WorkloadNotOK" AS (
SELECT DISTINCT "ImageWorkload"."idWorkload" FROM "InputImage", "
    ImageWorkload" WHERE NOT EXISTS (SELECT "t".* FROM "OUTPUT" AS "t"
    WHERE "InputImage"."imageName" = "t"."origName") AND "InputImage"."id" = "
    ImageWorkload"."idImage" ))
```

A questo punto, selezioniamo, come nel caso precedente, le immagini in base al costo, escludendo però quelle per le quali il workload non può essere migrato.

```
CREATE VIEW "OUTPUTNOWORKLOAD" ("origName","offerName","numCPU", "
    physical/virtual", "osNameOrig", "osNameTarget", "osMigrationEffort", "#softwares", "
    workload")
AS ( SELECT DISTINCT "InputImage"."imageName", "Offer"."imageName", "Offer"."
    numCPU", "InputImage"."physical", "InputImage"."osName", "Offer"."osName", "
    OSUpgrades"."effort", (SELECT COUNT(*) AS softwareNumber FROM "
    InputSoftware" WHERE "InputImage"."id" = "InputSoftware"."idImage"),
```

```
"ImageWorkload"."idWorkload"
FROM "InputImage" JOIN "ImageWorkload" ON "InputImage"."id" = "ImageWorkload"."
    idImage", "Offer", "OSUpgrades" WHERE "InputImage"."osBits" = "Offer"."osBits"
    AND ("InputImage"."osName" = "OSUpgrades"."originalOSName" AND "
    OSUpgrades"."destinationOS" = "Offer"."osName") AND "InputImage"."numCPU" <=
     "Offer"."numCPU" AND "ImageWorkload"."idWorkload" NOT IN (SELECT "
    idWorkload" FROM "WorkloadNotOK")
```

La query di selezione, infine, sarà analoga a quelle viste poco sopra.

```
SELECT "t".* FROM "OUTPUTNOWORKLOAD"
AS "t" INNER JOIN ( SELECT "origName", MIN("cost") AS minCost FROM "
    OUTPUTNOWORKLOAD" GROUP BY "origName" ) "smaller" ON "t"."cost" = "
    smaller".minCost AND "t"."origName" = "smaller"."origName"
```

5.6 Test riguardanti l'analisi statica

Durante l'attività di tesi sono state svolte delle attività reali di supporto all'analisi della migrazione. Tra questi casi, presentiamo prima le caratteristiche dei dati in input di un primo caso, per capire un ambiente tipico che ci si trova ad affrontare in ambito enterprise, mentre poi analizziamo in profondità un secondo caso, in modo da poter valutare l'efficienza dello strumento proposto.

5.6.1 Input del primo caso di studio

Questo caso di studio comprende i sistemi informativi di una azienda di dimensioni piuttosto grandi, che ha già concluso con successo un processo di virtualizzazione e che, come evoluzione successiva, desidera esplorare i vantaggi che gli verrebbero offerti in un ambiente di tipo cloud.

Caratteristica	#	Percentuale
RAM elevata	80	8,2%
Numero CPU elevato	122	12,6%
Entrambi	7	0,7%
Altre	762	78,5%

Tabella 5.1: Caratteristiche hardware

Dal punto di vista hardware, su un campione significativo di 971 immagini, po-

trebbero porre dei problemi nel passaggio al cloud quelle con requisiti particolari come numero alto di processori e/o quantità elevate di RAM, come si può vedere in tabella 5.1; tali limiti possono essere un problema per applicazioni di calcolo numerico oppure, soprattutto nel caso della memoria, per macchine utilizzate come database.

Sistema operativo	#	Percentuale
Windows	606	62,4%
Red Hat Linux	35	3,6%
AIX	261	28,9%
HP-UX	16	1,6%
Altri Linux	15	1,5%
Non disponibile	18	1,8%

Tabella 5.2: Sistemi operativi

Per quanto riguarda i sistemi operativi adottati, invece, sullo stesso campione abbiamo ottenuto i risultati visibili in tabella 5.2, che ci dicono come l'avere a disposizione sull'ambiente cloud di sistemi operativi Linux di fascia enterprise e Microsoft Windows ci permetta di coprire più della metà dei sistemi originali. Resta scoperta la fascia di sistemi su piattaforma AIX, il cui supporto è comunque previsto nel caso di IBM con la soluzione SCE+. Tali risultati, anche se con percentuali diverse, sono stati confermati in altri casi che abbiamo affrontato.

5.6.2 Descrizione del secondo caso di studio

Questo caso di studio prevedeva l'analisi della possibilità di migrazione di un cliente del settore bancario verso un ambiente cloud. Il campione di partenza, raccolto tramite strumenti automatici, era costituito da 894 immagini, per cui, come nella situazione precedente, il ricorso a sistemi automatici di analisi, quali quello sviluppato nel corso della nostra tesi, è risultato indispensabile.

Le immagini sono state rese disponibili in un foglio di calcolo Excel secondo un formato ben definito e la prima operazione effettuata è stata quella di riconciliazione dei dati ottenuti dal cliente in fasi diverse, che a volte presentano informazioni contrastanti o incomplete. Una volta consolidati i dati da cui partire, essi sono stati forniti in ingresso al nostro parser semplificato, il quale ne ha scartate 38 per l'assenza di informazioni sul sistema operativo dell'immagine, che a nostro giudizio, assieme al numero di bit, è un requisito minimo per poter procedere ulteriormente con l'analisi.

Sulla base delle immagini rimanenti, possiamo effettuare alcune considerazioni sui sistemi operativi presenti: un'ampia maggioranza del nostro campione è costituita da macchine dotate di un sistema operativo di casa Microsoft più o meno recente, mentre il resto consiste in gran parte in versioni differenti di Unix; solo una piccola minoranza, infine, era dotato di un sistema operativo Linux, in questo caso la distribuzione Red Hat. Il 66% circa dei sistemi operativi, inoltre, era a 32 bit, mentre il resto era a 64 bit.

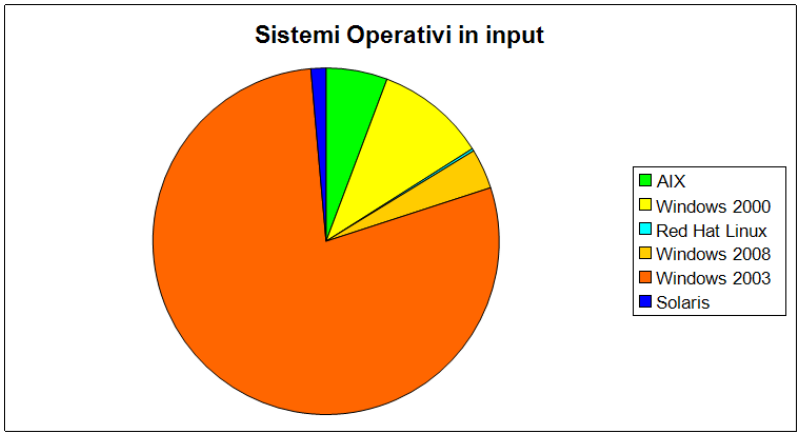

Figura 5.4: Sistemi operativi in input

5.6.3 Immagini prodotte in output

La prima operazione fatta è stato, come menzionato poco fa, un controllo della correttezza dei dati, che nel nostro caso ha scartato 38 immagini per l'assenza del sistema operativo; le 856 immagini rimanenti sono state invece caricate nel database per l'ulteriore analisi.

Il passo successivo è quello di applicare una delle query previste per selezionare le immagini in output desiderate; tra le query possibili, nel nostro caso abbiamo applicato quella presentata in sezione 5.5.1, la meno restrittiva dal punto di vista dei parametri presi in considerazione, ottenendo 821 immagini; ciò significa che sono stati segnati come non pronte per la migrazione 35 immagini e le cause di questa esclusione sono state riassunte nel grafico 5.5.

Abbiamo poi preso in considerazione anche l'analisi dal punto di vista dei workload: in questo caso i workload esclusi dalla prima analisi erano 13 e la loro esclusione ha provocato la conseguente eliminazione dal risultato di altre 107 immagini, che risultavano collegate

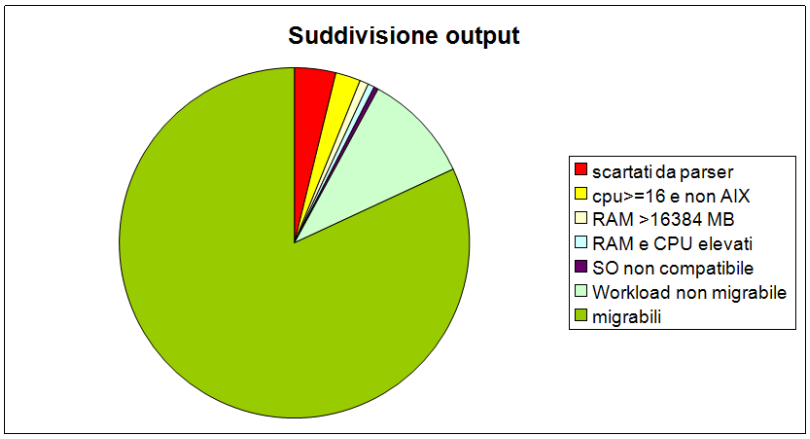

Figura 5.5: Risultati analisi CloMAs

ai workload esclusi. Dunque, nel caso di una visione combinata dei workload, verrebbero indicate come pronte per la migrazione 714 immagini tra tutte quelle fornite in input.

5.6.4 Tempi impiegati

Come si è detto, sono supportati due database come backend, uno completamente locale basato su HSQLDB e uno distribuito che sfrutta DB2, realizzato nei nostri test per mezzo di una macchina virtuale messa a disposizione dalla soluzione IBM Smart Cloud Enterprise. Come è evidente, le due soluzioni, pur producendo gli stessi risultati, hanno prestazioni notevolmente diverse e vedremo di seguito nei test come il fattore principale che incide sulle prestazioni sia la latenza e le prestazioni delle connessioni.

I test che si appoggiano sul database DB2 sono stati effettuati su due datacenter diversi, in modo anche da avere un valore indicativo di quanto tale scelta, al di là delle problematiche legali che possono orientare la decisione, possa incidere sulle prestazioni. I datacenter scelti sono stati quelli di Ehningen in Germania e quello di Raleigh, localizzato nel sud est degli Stati Uniti e le prove sono state fatte in orari della giornata diversi per cercare di ridurre la possibile incidenza di questo aspetto sui risultati.

Va fatto notare come questi test, essendo effettuati a partire da macchine non in cloud, abbiano risultati piuttosto variabili; nel corso della fase di analisi dinamica proporremo, invece, una modalità che ci permette di valutare le prestazioni delle connessioni in maniera più affidabile e meno soggetta a perturbazioni momentanee nella connessione tra l'utente

e il datacenter di destinazione.

5.6.4.1 Popolamento del database

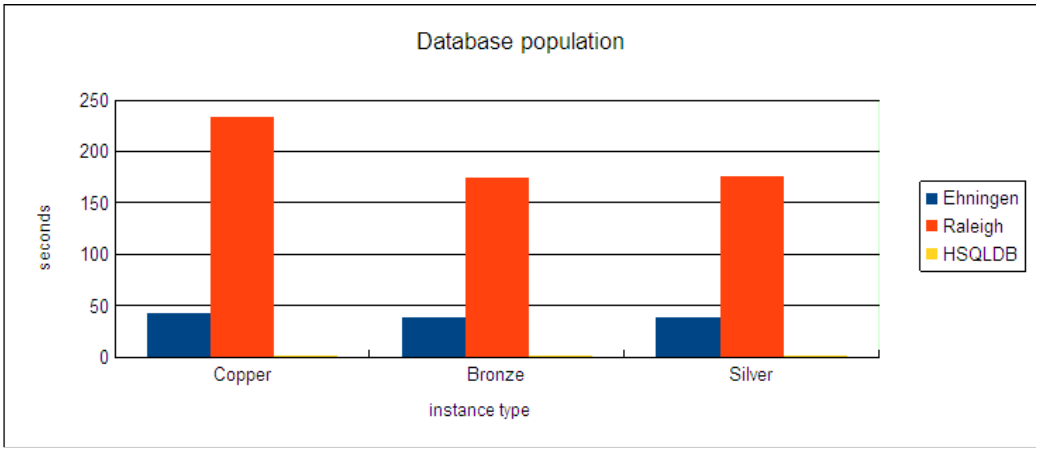

Figura 5.6: Tempi richiesti per il popolamento del database

Questa fase è quella iniziale e comprende, nei nostri test, sia le operazioni di parsing dei file di input che l'algoritmo di calcolo delle relazioni tra immagini e workload, dandoci un'idea del tempo dopo il quale l'utente può iniziare la fase di matching vera e propria. In questo caso si sono viste grandi differenze tra l'uso di un database locale e di uno remoto, principalmente a causa del grande numero di query da eseguire, necessarie per costruire le *foreign key* all'interno del database, ognuna delle quali causa una latenza. Notevoli differenze sono anche emerse tra i due datacenter, il che potrebbe essere una discriminante nel caso le latenze siano importanti.

5.6.4.2 Algoritmo per ottenere le relazioni tra immagini e workload

Abbiamo tenuto traccia dei tempi necessari per l'esecuzione dell'algoritmo 1, il quale, su un laptop non recentissimo come quello su cui sono stati lanciati i test – dotato di processore Intel Core2 Duo P8600 e 2GB di RAM – ha prodotto i risultati in pochi millisecondi; per questo motivo i tempi non sono stati riportati in quanto non particolarmente significativi.

Da ciò possiamo concludere come per una quantità di dati come quella affrontata, che rappresenta una situazione media che l'utente si troverà ad affrontare, l'algoritmo si comporta in modo adeguato.

5.6.4.3 Query per la selezione delle immagini da migrare

I test sono stati fatti sulla query vista in sezione 5.5.1, che è stata considerata quella di base per il nostro sistema. In questo caso viene analizzato un discreto numero di righe, quindi, similmente al caso del popolamento del database, abbiamo osservato delle differenze abbastanza rilevanti, mentre sono stati omessi i risultati ottenuti con il database HSQLDB, poiché risultavano molto bassi.

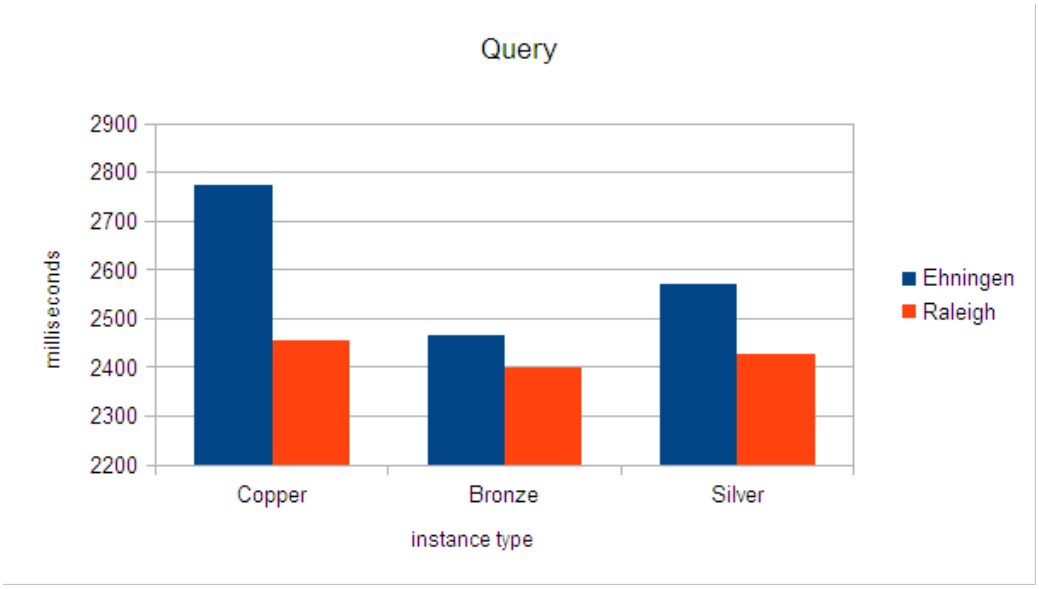

Figura 5.7: Tempi richiesti per l'esecuzione della query

Risulta interessante notare come non ci siano miglioramenti passando da un'istanza di tipo bronze a una silver, ma anzi un lieve peggioramento, che va considerato come un errore di misura. Questo risultato indica che, probabilmente, il passaggio a un'istanza superiore non garantisce miglioramenti, in quanto le risorse non vengono saturate, per cui, in un caso del genere, l'istanza migliore, anche in base al prezzo richiesto, è quella di tipo bronze.

Possiamo quindi concludere come, in questo scenario, il collo di bottiglia sarà sicuramente nelle elaborazioni e nel popolamento del database che conserva i dati, in particolare se esso è situato in remoto. I tempi richiesti per le elaborazioni locali, invece, sono gestibili senza troppi problemi.

5.7 Analisi dinamica

Per studiare l'analisi dinamica, a causa di problematiche organizzative, non è stato possibile fare degli esperimenti su un caso reale. È stato invece approntato un caso che modellizza uno scenario emerso frequentemente durante il periodo di tesi, cioè un'applicazione multitier, soluzione che, in scala più o meno grande, viene richiesta da ogni tipo di azienda, per esempio per ospitare il proprio sito Web o altri applicativi, anche ad uso interno. Faremo di seguito delle considerazioni generali su ambienti di questo tipo, per poi descrivere la soluzione specifica sottoposta a test, con i risultati ottenuti.

5.7.1 Automazione

Al momento, il procedimento proposto per l'analisi dinamica è composto da passi riproducibili, ma da eseguire in genere in modo manuale. Nell'ottica di una possibile automazione delle diverse fasi, elemento che è molto importante se ci troviamo a lavorare su grandi quantità di immagini, si potrebbe agire sia nella fase di importazione sull'ambiente cloud che nella fase di stress test, come riassunto qui di seguito.

5.7.1.1 Importazione

Per quanto riguarda l'importazione, la strada della reinstallazione dell'intero stack software è sempre possibile ed è stata utilizzata nell'attività di test descritta di seguita. Tuttavia, questa strada è percorribile solo in presenza di piccoli numeri; in alternativa, in presenza di strumenti automatici e di API, è possibile importare delle immagini virtuali preesistenti. In questo modo si potrà passare direttamente alla correzione delle configurazioni, con evidenti risparmi dal punto di vista dei tempi ed evitando anche gli errori che possono nascere da operazioni manuali ripetitive.

La possibilità di un procedimento di questo tipo è tuttavia strettamente legata alle offerte dai singoli provider cloud, per cui sarà necessario esplorare la documentazione fornita per verificare anche eventuali limiti, ad esempio sui sistemi operativi supportati.

5.7.1.2 Load Balancing automatico

Lo sviluppo ulteriore, per quanto riguarda il nostro algoritmo, è quello di avere un load balancer dinamico. Questo è infatti uno dei vantaggi principali di un ambiente cloud, il

quale, grazie alla rapidità con cui è possibile creare nuove immagini, permette di reagire in pochi minuti a eventuali picchi.

Infatti, il nostro algoritmo prevede una fase di test e una di misurazione delle prestazioni con eventuali correzioni delle configurazioni, nel caso esse non siano soddisfacenti. Nel nostro scenario queste operazioni vengono effettuate prima dell'esecuzione vera e propria, mentre, se le applichiamo al momento dell'esecuzione, otteniamo appunto un load balancer dinamico.

Per questo motivo i tempi da dedicare all'analisi dinamica non devono essere molto elevati, ma devono soprattutto dare un'idea di massima delle prestazioni in condizioni normali. Non sarà invece necessario preoccuparsi, al momento della migrazione, dell'eventualità che si manifestino picchi di richieste e della loro gestione, in quanto situazioni di questo tipo possono essere poi gestite dinamicamente: le prestazioni vengono costantemente monitorate, in base a criteri scelti dall'utente e, in caso di necessità, è possibile aggiungere delle istanze o eliminarne nell'arco di pochi minuti.

5.7.1.3 API standard

Nell'ottica di ridurre i tempi spesi in operazioni ripetitive, risulta utile valutare l'utilizzo, anziché delle interfacce web di controllo previste da tutti i provider cloud, di API nel linguaggio preferito dall'utilizzatore. Tale scelta è facilmente perseguibile, se si desidera effettuare le prove su un singolo ambiente di destinazione, in quanto ogni provider mette a disposizione degli strumenti di questo tipo, mentre, nel caso più interessante di confronto tra ambienti differenti, nasce l'esigenza di sfruttare API standard che astraggano dalle peculiarità ed evitino per quanto possibile problematiche di lock-in.

Esistono diverse API di questo tipo, che si differenziano principalmente per il linguaggio di programmazione in cui sono scritte e il numero di provider supportati. Da quest'ultimo punto di vista, il limite principale riscontrato nel corso dei nostri studi è nel supporto offerto per la soluzione IBM oggetto principale dei test, che, non godendo ancora di una diffusione molto ampia, non viene gestita in modo completo da diverse API di questa categoria.

L'altro limite di tali API è intrinseco nell'idea stessa di unificazione: esse cercano infatti di fornire un modello comune per tutti gli ambienti, il che costringe ad avere un nucleo minimo di operazioni comuni; tutte le operazioni più avanzate (per esempio quelle

legate all'importazione di immagini preesistenti, per le quali c'era un particolare interesse nel corso dell'attività di tesi) non sono presenti o vengono al più esposte come estensioni specifiche di ogni provider, ma non sono offerte per il prodotto che è l'oggetto delle nostre prove.

Fra le API standard, abbiamo esplorato in particolare l'utilizzo di `libcloud`, scritta in linguaggio Python e che offre un buon supporto per SCE. Visto che tale API non supporta, per il momento, la creazione di immagini a partire da istanze attive, si è optato per creare prima le macchine ed eseguire tutte le configurazioni necessarie, salvarle come immagini tramite il portale IBM per poi avviare le varie istanze oggetto dei test ed eliminarle alla conclusione.

A questo scopo presentiamo di seguito uno script che permette all'utente di creare in modo rapido delle istanze a partire dalle immagini approntate. L'utente dovrà specificare tre soli parametri: il nome dell'immagine dalla quale partire, il tipo di istanza da creare e il datacenter scelto. Lo script termina quando l'istanza è stata creata con successo e verrà fornito anche l'indirizzo IP assegnato dal provider, in modo da consentire l'accesso.

Il grande vantaggio permesso dall'utilizzo di API standard consiste nella possibilità di adattarle a uno qualsiasi dei provider supportati, senza necessità di effettuare modifiche alla parte funzionale: sarà infatti sufficiente scegliere uno dei driver inclusi per cambiare il provider utilizzato, mentre tutto il resto può rimanere invariato, con conseguente notevole risparmio di tempo, per esempio nel caso di test contemporanei su molteplici fornitori.

Tale script potrebbe essere la base di uno strumento di load balancing dinamico, discusso brevemente nella sezione precedente. In questo caso sarebbe necessario aggiungere un componente in grado di raccogliere i dati sulle prestazioni delle macchine in tempo reale, o installando degli appositi sensori sulle macchine o tramite gli strumenti disponibili nel caso si abbia il controllo sull'hypervisor sottostante e dunque una visione di basso livello. Sulla base di tali dati, lo script può quindi creare o eliminare istanze, nel caso più interessante anche su provider cloud differenti, in base a criteri decisi in precedenza e reagire così a possibili picchi in un tempo che è limitato esclusivamente da quello richiesto per la creazione di un'istanza. Strumenti di questo tipo sono forniti già pronti da alcuni fornitori dell'ambito IaaS, fra i quali segnaliamo Amazon con Elastic Load Balancer e Rackspace Cloud Load Balancers.

```
1 #!/usr/bin/env python
2 """createInstance.py: Creates an instance based on a previously created image"""
```

```
3
4  from libcloud.compute.base import NodeAuthSSHKey
5  from libcloud.compute.types import Provider
6  from libcloud.compute.providers import get_driver
7  import libcloud.security, time
8
9  #Credentials
10 ACCESS_ID = ''
11 SECRET_KEY = ''
12
13 #Connect to the provider
14 Driver = get_driver(Provider.IBM)
15 conn = Driver(ACCESS_ID, SECRET_KEY)
16 #Use this key for SSH access. Windows instance will need userName and password
17 k = NodeAuthSSHKey('ChiaveProva')
18
19 wantedImage = raw_input('Image name: ')
20 instanceName = raw_input('New instance name: ')
21 sizes = conn.list_sizes()
22 for index, item in enumerate(sizes):
23     print index, '-->', item.name
24 instanceType = int(raw_input('Choose instance type number (from 0 to ' + str(len(sizes)
       -1) + '): '))
25 locations = conn.list_locations()
26 for index, item in enumerate(locations):
27     print index, '-->', item.name
28 chosenLocation = int(raw_input('Choose location number (from 0 to ' + str(len(locations)
       -1) + '): '))
29
30 #List all images
31 images = conn.list_images()
32
33 #Create instance
34 for image in images:
35     if wantedImage == image.name:
```

```
36          print 'Found image ', image
37          node = conn.create_node(name=instanceName, image = image, size=sizes[
                instanceType], location=locations[chosenLocation], auth=k)
38
39  #Wait till the image it's ready
40  while node.state != 0:
41      print 'Still waiting...'
42      time.sleep(30)
43      nodes = conn.list_nodes()
44      for node in nodes:
45          if node.name == instanceName:
46              break
47  #Then print IP assigned to the new instance
48  print instanceName + ' ready at ' + node.public_ips[0]
```

5.7.2 Modalità di deployment

Per una soluzione di questo tipo, esistono diverse modalità di deployment che si possono classificare in modo progressivo per la scalabilità offerta; d'altra parte, all'aumentare della scalabilità, aumenta anche la complessità generale e, in particolare, quella di installazione e gestione delle diverse soluzioni.

5.7.2.1 Macchina singola

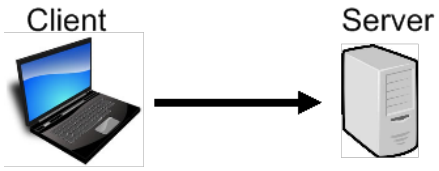

Figura 5.8: Macchina singola

La prima soluzione possibile è quella di avere un'unica macchina che contenga sia il web server che il database a cui esso si appoggia. Questa configurazione è facile da installare e configurare, ma d'altro canto la scalabilità offerta è molto bassa: l'unica possibilità è

infatti quella di incrementare la potenza della macchina. Sebbene questa soluzione sia perseguibile in un ambiente *on premises*, dove è possibile ampliare per esempio la RAM sfruttando le diminuzioni continue dei prezzi dei componenti, in ambito cloud, dove i margini di incremento devono sottostare al catalogo previsto, si rischia di raggiungere ben presto il limite massimo. La ridondanza offerta è nulla: in caso di problemi alla macchina che fornisce il servizio, esso non sarà disponibile fino al ripristino della macchina stessa.

5.7.2.2 Database su una macchina diversa

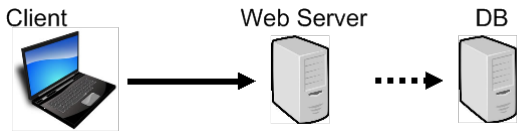

Figura 5.9: Database separato

La prima evoluzione rispetto alla soluzione precedente consiste nello spostamento del database su una macchina a sé stante. Questo migliora la scalabilità senza complicare eccessivamente il setup, anche se l'unica modalità prevista è sempre quella verticale, che in questo caso può essere fatta sia sulle macchine adibite a web server sia su quelle che gestiscono il layer dei dati. La ridondanza rimane sempre un punto critico nel caso di problemi su una delle due macchine.

Dal punto di vista delle prestazioni ci può essere un peggioramento rispetto alla situazione precedente, vista la latenza che viene introdotta nel collegamento tra le due macchine.

5.7.2.3 Load Balancer sul web server

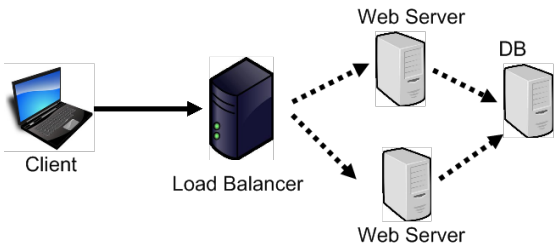

Figura 5.10: Aggiunta di un load balancer sui web server

Nell'ottica di un'evoluzione verso la scalabilità orizzontale, l'idea più semplice è quella di bilanciare il frontend e quindi far passare tutte le richieste attraverso un load balancer, che le smisti poi ai diversi web server, i quali si interfacceranno con un singolo database. Questa struttura permette un buon aumento della scalabilità e anche della ridondanza sul web server, mentre il database rimane un singolo punto di possibile fallimento e rischia ben presto di diventare il collo di bottiglia per esaurimento delle risorse o problematiche di lock, dipendenti anche dal prodotto utilizzato. È possibile pensare di aumentare ulteriormente la ridondanza, raddoppiando il load balancer e sfruttando, per esempio, una replicazione master slave, nel quale lo slave potrebbe essere attivo e monitorare costantemente il master, prendendone il posto in caso di caduta.

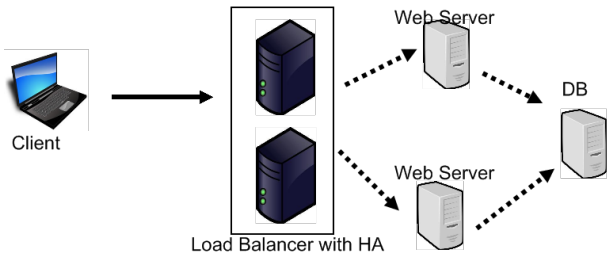

Figura 5.11: Load balancer in *high availability*

Per quanto riguarda le prestazioni, si introduce un'ulteriore latenza per la necessità di passare dal load balancer; tuttavia, in questo caso i tempi di collegamento tra il load balancer e i web server sono meno critici rispetto a quelli tra i web server e il rispettivo database.

5.7.2.4 Bilanciamento del database

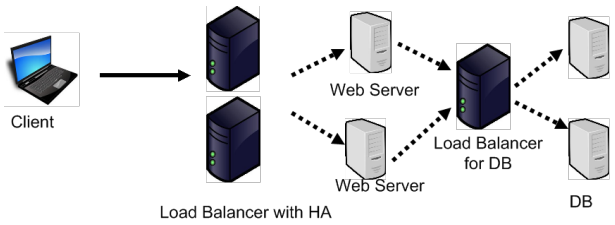

Figura 5.12: Bilanciamento del database

Al termine di queste evoluzioni rimane un importante collo di bottiglia nel database. Una prima possibile soluzione è quella di creare uno strato di caching, in grado di ridurre il numero di accessi al database: in caso di necessità viene quindi prima verificata la presenza dei dati nella cache e solo in caso negativo o nel caso esso sia presente, ma non più valido, esso viene recuperato dal database.

Ciò però è solo un palliativo, mentre la soluzione più adeguata a lungo termine sembra quella di porre più istanze in parallelo del database o attraverso un proxy oppure con una struttura a cluster. Va fatto notare come i tradizionali database relazionali rendano complicato uno scenario di questo tipo, soprattutto se l'ottica è quella di dividere i dati sulle varie macchine (*sharding*), in quanto nascono problemi per quanto riguarda il rispetto delle proprietà ACID ed è necessario ricreare su ogni nodo l'insieme delle relazioni.

Per questo, in uno scenario cloud, soprattutto per applicazioni che debbano gestire moli molto grandi di dati, molte aziende stanno migrando da soluzioni relazionali, che soprattutto nel caso di startup sono frequentemente basate su MySQL, verso soluzioni non relazionali, raggruppate sotto il nome di NoSQL. Per fare un esempio, il social network Twitter ha dovuto esplorare soluzioni di questo tipo in quanto già nel 2010 doveva gestire 7 TB di nuovi dati al giorno; tale mole di dati ha richiesto interventi sia dal punto di vista della memorizzazione sia soprattutto da quello delle interrogazioni, che diventavano rapidamente ingestibili nel caso di dati distribuiti su milioni di righe. Per risolvere questi limiti sono stati introdotte via via soluzioni di query basate su Hadoop – basate sul paradigma MapReduce – e il database distribuito Cassandra, sviluppato in origine presso Facebook[3].

Tali soluzioni, al prezzo di riduzioni del livello di consistenza permesso, automatizzano e rendono molto più semplice la scalabilità orizzontale su diverse macchine; per una trattazione approfondita dal punto di vista teorico della tematica NoSQL rimandiamo a [Str10].

5.7.3 Caso di studio

L'ambito delle applicazioni web multi-tier è molto vasto e ramificato, con la presenza di numerosi framework e middleware che si contendono le quote di mercato; tra di essi possiamo menzionare le soluzioni basate su PHP, quelle su J2EE, il framework .NET, Ruby on Rails e altre ancora. All'interno di questo ambito abbiamo scelto di effettuare

[3] http://www.slideshare.net/kevinweil/nosql-at-twitter-nosql-eu-2010

dei test basati su un'installazione di Drupal[4], un Content Management System opensource che si posiziona stabilmente tra i primi 3 usati al mondo[5] e che dal punto di vista tecnico sfrutta il diffuso stack LAMP (dalle iniziali degli elementi alla base, ossia Linux come sistema operativo, Apache Web Server, MySQL come database e infine PHP o Perl come interprete dinamico). Drupal spicca inoltre per avere un'ampia comunità di sviluppatori e viene utilizzato per siti di ogni dimensione, fra i quali citiamo la pagina Web del presidente degli Stati Uniti d'America.

Il software utilizzato è alla versione 7.10 e, all'installazione di base con supporto per MySQL, è stato aggiunto il modulo `devel`, scaricabile a parte dal sito della comunità Drupal. Esso ci ha permesso di popolare automaticamente il database con 50 utenti e 100 articoli generati in maniera casuale, in modo da simulare i contenuti iniziali di un piccolo sito a cui gli utenti accedono per consultazione ed eventualmente per modifiche. Ipotizziamo inoltre che la configurazione di tutti i componenti coinvolti sia già stata ottimizzata, anche se questo parametro andrà tenuto in considerazione in uno scenario reale, poiché potrebbe avere una grossa influenza sulle prestazioni complessive.

L'applicazione è stata testata sulla soluzione SCE, sia dal punto di vista della scalabilità verticale che di quella orizzontale; a differenza dei test sull'analisi statica, le richieste sono inviate da un'istanza che viene creata nello stesso datacenter delle altre macchine, così da evitare di prendere in considerazione le variazioni dovute alla connessione tra l'utente finale e il datacenter, che sono difficilmente quantificabili poiché dovute a un numero di fattori molto ampio. In questo modo possiamo produrre dei risultati riproducibili e confrontabili.

Le richieste sono destinate a pagine di contenuto diverse del sito Drupal e bilanciate tra quelle fatte da utenti autenticati, che creano un carico maggiore dal punto di vista delle risorse di sistema, in quanto richiedono accessi al database per la gestione delle risorse, e da utenti non autenticati, che non possono invece usufruire nella configurazione di base di Drupal di funzionalità quali la ricerca, ma che normalmente costituiscono la maggior parte degli accessi. Per lanciare le richieste e raccogliere i risultati, è stata utilizzata l'applicazione opensource JMeter [AJP12], che ci consente anche di gestire elementi avanzati quali i cookie di sessione e, una volta impostati i tipi di richieste desiderati, permette di replicarle più volte con eventuali timer e pause calibrate tra ogni gruppo di richieste. Esistono numerosi software commerciali che possono essere usati in alternativa a questo

[4] http://drupal.org/
[5] http://trends.builtwith.com/cms

scopo; per esempio IBM propone il software Rational Performance Tester, in grado di offrire funzionalità avanzate, quali il discovery delle relazioni tra i dati delle applicazioni.

Presentiamo di seguito i risultati che sono stati ottenuti su queste soluzioni, mettendo in risalto le prestazioni e concludendo con una breve analisi del lato economico.

5.7.3.1 Criteri di valutazione

Come si è detto, i criteri di valutazione per questo tipo di analisi vanno valutati caso per caso. Per un'applicazione Web, il parametro più utile è quello che ci permette di valutare il comportamento in base al numero di richieste ricevute, così da vedere la quantità di utenti che è possibile servire mantenendo i criteri di qualità richiesti. L'aumento del numero di utenti consente anche di verificare la *graceful degradation*: il sistema deve comportarsi in maniera predicibile anche superando il numero massimo di utenti che può servire, fornendo eventualmente loro dei messaggi di errore adeguati.

Di seguito, presenteremo gli scenari presi in considerazione, soprattutto sulla base del numero di utenti che esso ci permette di servire e i tempi di risposta medi previsti nel caso del nostro scenario. Ovviamente, un test del genere offre risultati tanto più validi quanto lo scenario (e quindi le richieste) che viene simulato rappresenta il comportamento reale degli utenti del sistema sottoposto a verifica, per cui sarà utile tenere traccia delle azioni e delle richieste reali o recuperarli a partire dalle statistiche di accesso all'applicazione Web, per poi replicarli al momento dei test.

5.7.3.2 Macchina singola

Questo è il caso più semplice, ma anche il più limitato, soprattutto in termini di scalabilità futura; infatti, se chiaramente non ci sono problemi di latenza, in quanto tutto è a bordo della stessa macchina, in questo caso dotata di sistema operativo Suse Linux Enterprise 11 SP1 a 32 bit, se le prestazioni non sono sufficienti l'unica possibilità offerta da un'architettura del genere è quella di scalare verticalmente, passando dunque a una macchina più potente.

Nei test effettuati abbiamo preso in considerazione tutti i tagli previsti dall'offerta IBM SCE, presentati con le rispettive caratteristiche in tabella 2.1. Possiamo innanzitutto sottolineare come il comportamento di istanze di tipo copper e bronze siano molto simili: ciò è facilmente giustificabile in quanto esse hanno lo stesso quantitativo di CPU e memoria,

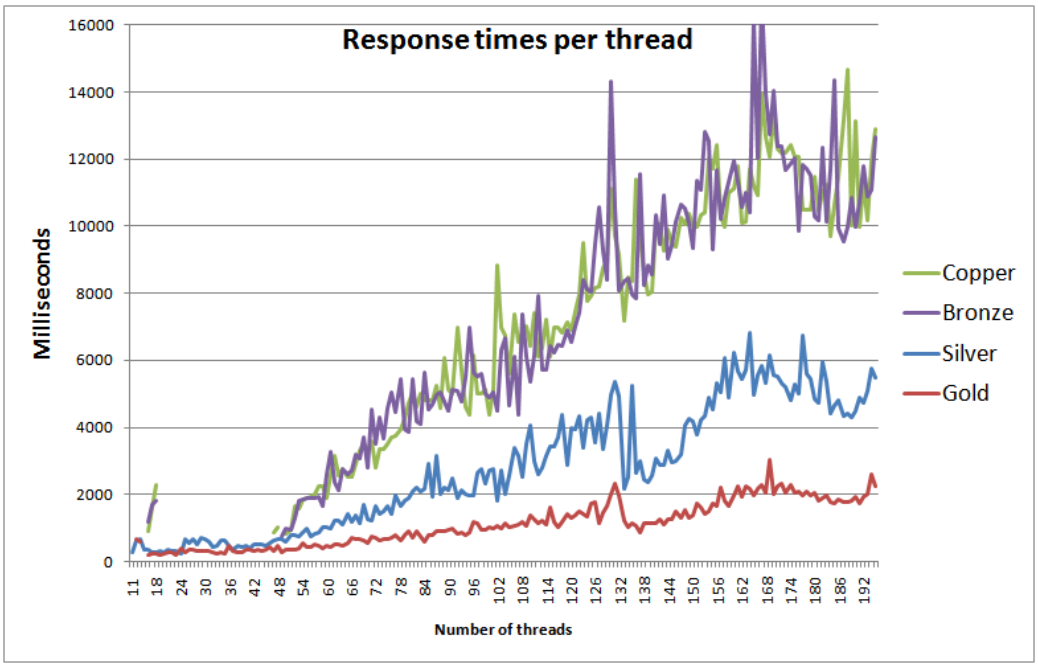

Figura 5.13: I tempi di risposta rilevati nella configurazione con macchina singola in base al numero di utenti contemporanei

mentre l'unico parametro che varia è lo spazio su disco. Miglioramenti sui tempi di risposta si hanno invece se aumenta il numero di CPU virtuali a disposizione (ognuna con frequenza di 1,25 GHz nel caso di SCE), che è il parametro che varia nel passaggio da un'istanza di tipo silver a una di tipo gold. Questo ci permette di concludere che il nostro case study ha un carico prevalentemente computazionale, mentre non ci sono limitazioni sulla memoria; essa potrebbe invece diventare risorsa scarsa nel caso si avesse un database di dimensioni più elevate.

Per quanto riguarda i tempi di risposta è inoltre da valutare quale sia la soglia massima accettabile, così da capire anche il numero di utenti che è possibile servire. Infatti, tempi superiori ai 5 secondi risultano già molto elevati in uno scenario reale, da evitare se si desidera garantire delle prestazioni adeguate agli utenti. Per questo, se ci vogliamo limitare per motivi di costi a istanze di tipo copper o bronze, sembrerebbe opportuno, in una situazione come quella di test non spingersi oltre a 70-80 utilizzatori contemporanei, pena il degrado progressivo dei tempi di risposta. Un'istanza di tipo gold è invece in grado di garantire tempi di risposta accettabili, inferiori ai 2 secondi, anche in presenza di più di 180 utenti nel sistema.

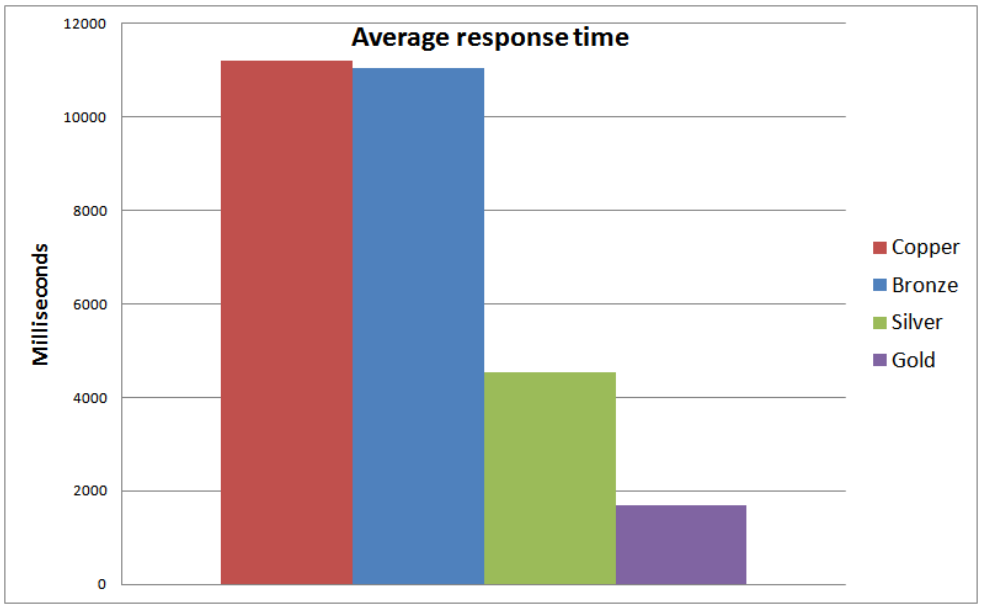

Figura 5.14: I tempi di risposta medi rilevati nella configurazione con macchina singola

5.7.3.3 Database su macchina dedicata

In questo caso si è spostato il database su una macchina virtuale a sé stante, sempre con sistema operativo SLES11 a 32 bit, così da disaccoppiare il layer dei dati da quello di presentazione e consentire anche un aumento delle possibilità di scalabilità. In una configurazione del genere, possiamo pensare di scalare verticalmente sia la macchina che ospita l'application server che quella che ospita il database, intervenendo prima sul layer che risulta maggiormente sotto stress: se quindi, come nel nostro caso, si deduce che il database non è sfruttato al 100%, possiamo pensare di investire maggiori risorse in una macchina più potente destinata all'application server.

I test delle prestazioni, visibili in figura 5.15 e 5.16, ci mostrano come, rispetto allo scenario di una macchina singola, si abbiano dei peggioramenti sui tempi medi nell'ordine del 5-10% a parità di tipo di istanza. Questo aumento dei tempi di risposta potrebbe essere dovuto sia al carico computazionale richiesto per contattare una macchina differente che ospiti il database, sia alla latenza di collegamento, che comunque dovrebbe essere minima visto che le istanze sono posizionate nello stesso datacenter. Nel caso specifico, registriamo dei picchi elevati – superiori ai 20 secondi – nel caso della configurazione meno potente, dovuta probabilmente a un eccesso di richieste che non possono essere gestite: durante

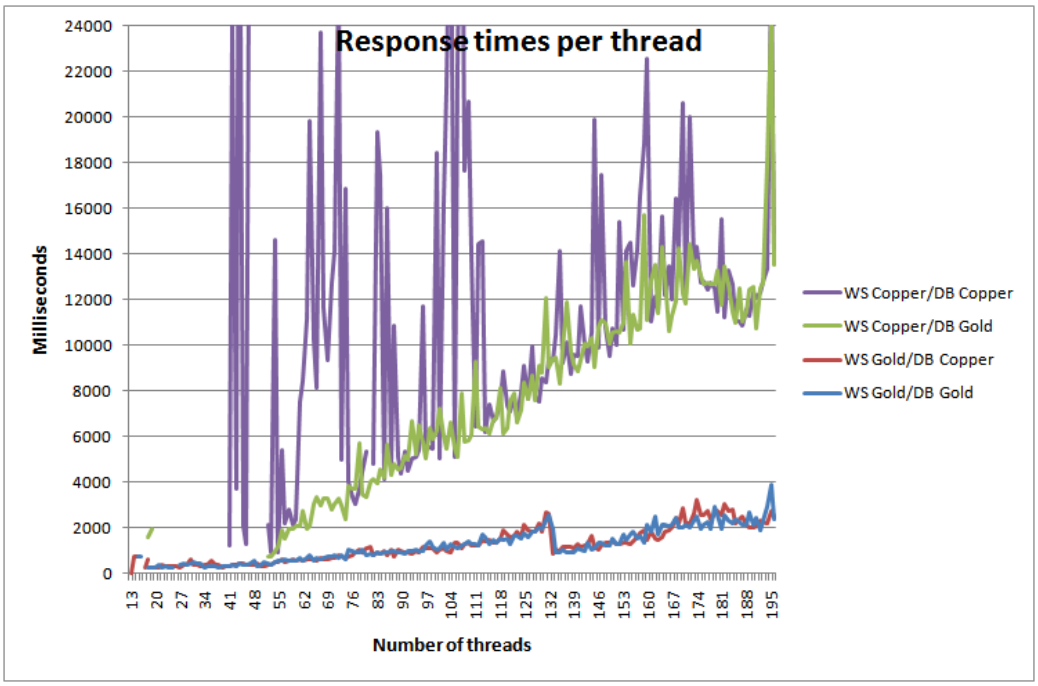

Figura 5.15: I tempi di risposta rilevati nella configurazione con database separato in base al numero di utenti contemporanei

i test abbiamo sempre rilevato un utilizzo totale della CPU. Invece, in uno scenario che prevede un web server di tipo gold, si riescono a gestire in tempi accettabili (inferiori a 2 secondi) anche un numero di richieste contemporanee piuttosto elevato.

Il secondo elemento che viene messo in evidenza, già emerso poco fa, è che nel nostro test case l'influenza della macchina che ospita il database sia molto bassa, per cui, in caso di necessità, sembra opportuno investire in un'istanza più performante esclusivamente per quanto riguarda il layer di presentazione.

5.7.3.4 Load balancer con test della scalabilità orizzontale

Se lo scenario precedente può risultare di utilità limitata e avere un costo maggiore di quanti siano i possibili vantaggi, esso assume maggior senso se lo vediamo come passo di base nello sviluppo di una strategia di scalabilità orizzontale.

Per avere ciò, è necessario introdurre un load balancer che smisti le richieste in base a uno degli algoritmi possibili, fra i quali quello più basilare e utilizzato nei nostri test sarà quello *round robin*, che assegna ogni richiesta a uno dei web server disponibili in modo ci-

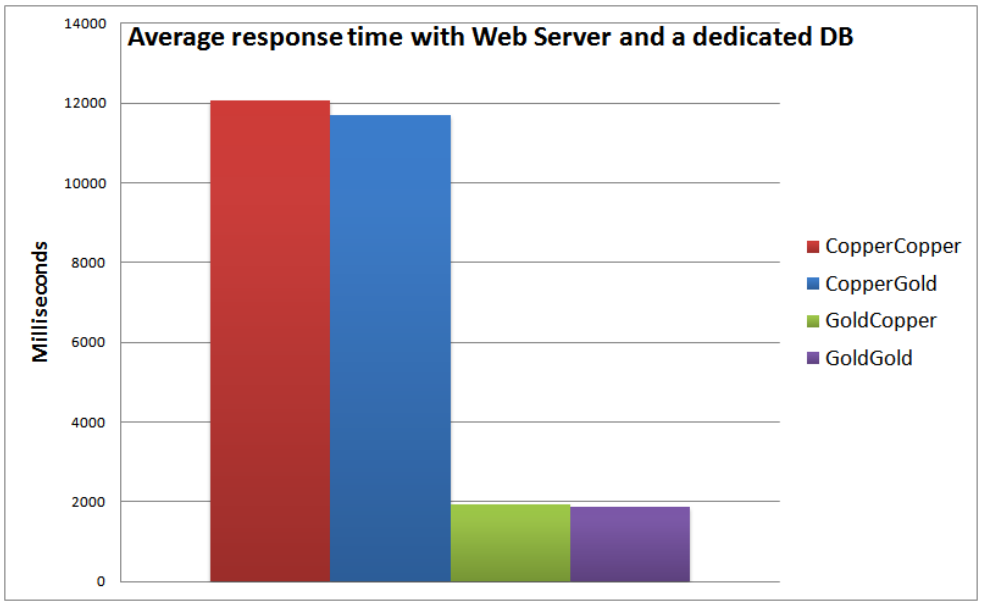

Figura 5.16: I tempi di risposta medi rilevati nella configurazione con database separato

clico, evitando fenomeni di *starvation*. I web server accederanno allo stesso database, che deve dunque essere posizionato su una macchina a sé stante, come nella configurazione precedente. Per il ruolo di load balancer viene impiegato il web server Cherokee [CP12], scelto per le buone prestazione fornite e per la facilità di configurazione, che è stato installato su una macchina Red Hat Enterprise Linux 6 a 64 bit; ovviamente, in applicazioni reali, si potranno utilizzare soluzioni equivalenti, per esempio basate su Apache (con il componente aggiuntivo mod_jk) o altre.

Il primo dato ricavato dai nostri test, riassunti in figura 5.17, è che, nel caso di un unico load balancer e unico web server, si introduce un ritardo nella risposta che è pari a circa il 33% rispetto al primo caso analizzato; inoltre, in questa configurazione, si è notato come un buon numero di richieste, circa il 17%, produca un errore in quanto il load balancer va in timeout, poiché l'unico web server a cui può reindirizzare le richieste è occupato e non può dunque gestirle. Questo timeout è ovviamente configurabile, ma è comunque spia di un rallentamento inaccettabile in un caso reale. Tale problema viene evidenziato anche nel grafico di figura 5.18, dove abbiamo i tempi di risposta in base al numero di utenti presenti nel sistema: nel caso di un unico web server il numero massimo di richieste da parte degli utenti, che non terminino con condizione di errore, è circa la metà di quelle consentite negli altri casi.

Questa configurazione è comunque scarsamente utile in un caso pratico, anche considerando l'aumento dei costi, pur permettendo di aumentare la ridondanza. Maggior interesse ha invece la configurazione con un secondo web server, con la quale si riesce a servire praticamente tutte le richieste, pur con un peggioramento dei tempi complessivi di risposta (siamo circa al 50% in più, rispetto a una configurazione dello stesso tipo senza load balancer). Infine abbiamo verificato il comportamento nel caso di aggiunta di un ulteriore web server, osservando dei miglioramenti interessanti: anche in questo caso si riescono a gestire tutte le richieste, ma i tempi sono comunque superiori del 15% circa a una soluzione priva di load balancer.

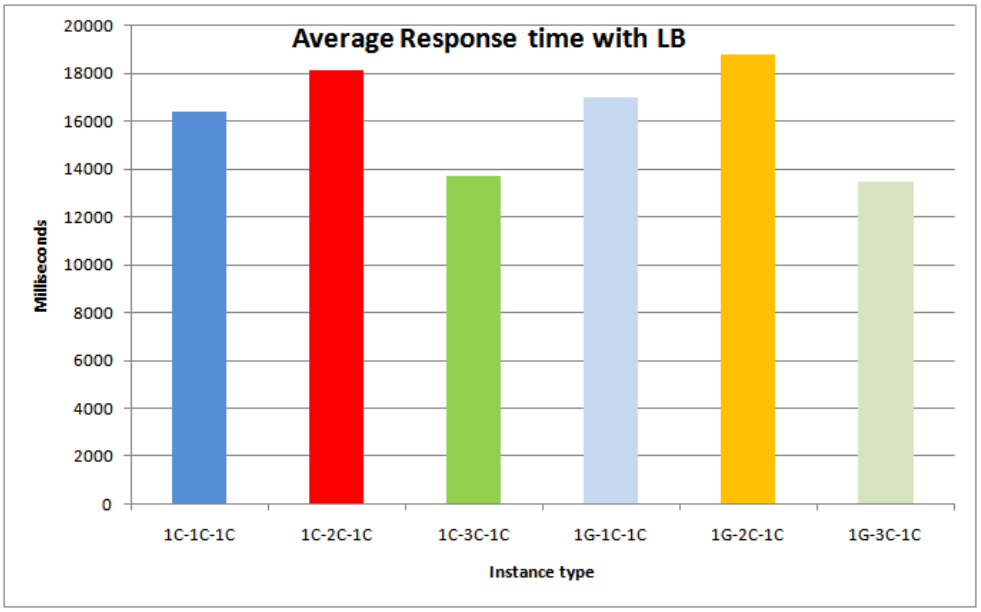

Figura 5.17: I tempi di risposta medi rilevati nella configurazione con load balancer

Dal punto di vista degli utenti serviti, questa soluzione, anche con tre webserver in parallelo, non garantisce prestazioni adeguate al di sopra dei 100 utenti contemporanei, con tempi di risposta che salgono rapidamento oltre i 2 secondi e mezzo; appare evidente come per gestire carichi di questo genere sia necessario utilizzare dei webserver più prestanti.

La seconda configurazione analizzata è stata quella equivalente alla precedente ma con un load balancer più potente (dotato nel nostro caso di 8 CPU virtuali anziché 2). Non sono stati rilevati cambiamenti significativi rispetto al caso precedente, se non dovuti a fattori temporanei, per cui, anche in questo caso, possiamo concludere come il collo di

bottiglia in un sistema di questo tipo sia localizzato nei web server. Questa conclusione viene anche confermata dall'analisi della percentuale di occupazione dell'istanza dedicata a load balancer, che si mantiene costantemente sotto l'1%, offrendo quindi ampi margini di aumento del numero di richieste. Visto che, dal punto di vista degli utenti serviti, le cifre non variavano in modo significativo rispetto a quelle riportate, questo scenario non viene mostrato in figura 5.18.

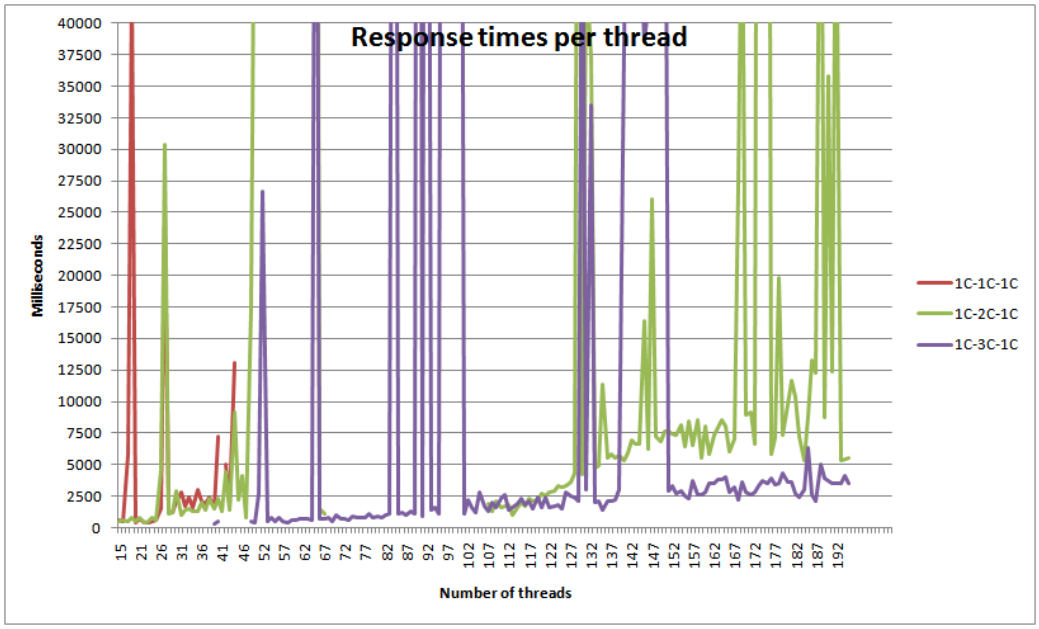

Figura 5.18: I tempi di risposta rilevati nella configurazione con load balancer e webserver di tipo copper in base al numero di utenti contemporanei

L'ultimo scenario analizzato nel corso dei nostri test è stato quello in cui a un load balancer poco potente (istanza di tipo copper) vengano associati uno o più server di tipo gold, dotati quindi di 4 CPU virtuali. In questo caso presentiamo i risultati dei test in base al numero di utenti attivi nel sistema e rileviamo come nel caso base, con un unico web server, i tempi siano di poco superiori alla configurazione senza load balancer, coerentemente con i risultati precedenti. Rispetto allo scenario appena visto, dove l'istanza più potente era quella del load balancer, i miglioramenti sono notevoli, il che ci conferma come nella configurazione in esame il fattore di maggior importanza e sottoposto a maggior carico sia il web server, sul quale vale dunque la pena di investire.

Se poi aggiungiamo degli ulteriori web server, si ottengono come atteso dei miglioramenti piuttosto significativi, che giustificano il maggior costo della soluzione: già con due web server in parallelo i miglioramenti raggiungono e superano il 50% e si riesce a gestire al meglio situazioni fino a 150 utenti attivi, che otterranno tempi di risposta intorno al mezzo secondo. Al di sopra di questo numero i tempi, pur rimanendo a nostro giudizio accettabili, iniziano a peggiorare e in uno scenario dinamico sarebbe consigliato l'aggiunta di un ulteriore web server, che garantisce, anche per questi carichi, tempi di risposta intorno ai 500 millisecondi.

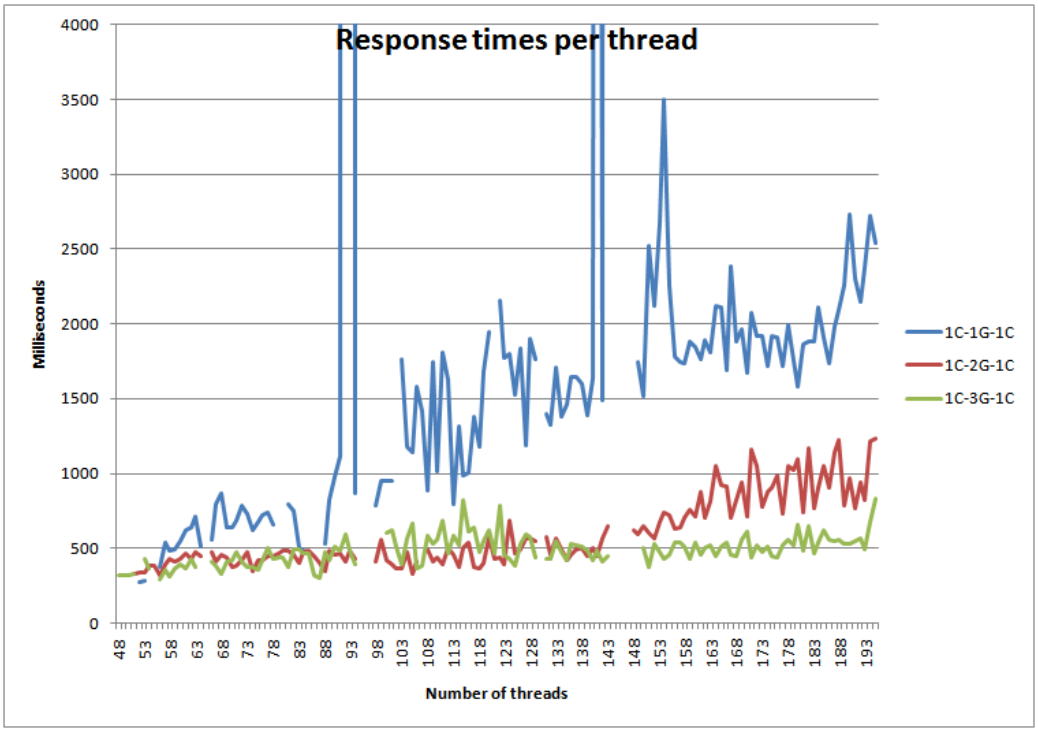

Figura 5.19: I tempi di risposta rilevati nella configurazione con load balancer e webserver di tipo gold in base al numero di utenti contemporanei

Per concludere, rimarchiamo la facilità con cui è possibile scalare in questa situazione: una volta predisposto la situazione di base, gli unici passi da compiere per aumentare la scalabilità sono la creazione di un web server aggiuntivo e la segnalazione della sua presenza al load balancer (che potrebbe eventualmente necessitare di un riavvio): dopo queste operazioni il sistema sarà in grado di migliorare le proprie prestazioni in maniera proporzionale. Una scalabilità verticale è invece meno rapida, in quanto è necessario creare

una nuova macchina, che poi vada a sostituire quella di potenza inferiore cercando nel frattempo di ridurre al minimo il *downtime* del sistema.

5.7.3.5 Costi

Grazie al modello dei costi a consumo, risulta piuttosto semplice calcolare il costo delle varie configurazioni e metterlo dunque in rapporto con le prestazioni ottenute. Mostriamo in tabella 5.3 i prezzi complessivi dei vari scenari presentati, sulla base del listino pubblico dei prezzi di Smart Cloud Enterpise valido a marzo 2012 e ipotizzando che ogni macchina rimanga costantemente accesa, per un totale medio di 720 ore mensili.

Nel caso di utilizzi sporadici, per altro piuttosto semplici da realizzare tramite strumenti di automazione come quelli presentati in sezione 5.7.1, si possono avere risparmi, più o meno consistenti, rispetto alle cifre qui riportate.

Load Balancer	Web server	Database	Costo complessivo
-	1 copper	-	33,12 €
-	1 bronze	-	39,6 €
-	1 silver	-	59,04 €
-	1 gold	-	92,88 €
-	1 copper	1 copper	66,24 €
-	1 copper	1 gold	126 €
-	1 gold	1 copper	126 €
-	1 gold	1 gold	185,76 €
1 copper	1 ÷ 3 copper	1 copper	141,84 ÷ 208,08 €
1 gold	1 ÷ 3 copper	1 copper	257,04 ÷ 323,28 €
1 copper	1 ÷ 3 gold	1 copper	218,88 ÷ 404,64 €

Tabella 5.3: Costi mensili delle configurazioni sottoposte a test

Da questi prezzi possiamo trarre alcune considerazioni generali: innanzitutto, è opportuno utilizzare web server prestanti, soprattutto se il numero di utenti medi che ci si aspetta è elevato.

Visti i costi, risulta poi particolarmente interessante una soluzione come l'ultima, magari partendo da una soluzione base con un unico web server che si comporta già in maniera adeguata per poi aggiungere eventualmente delle istanze in parellelo in maniera dinamica. In questo modo si riesce a sfruttare al meglio i tempi rapidi di creazione delle macchine consentiti da una soluzione cloud.

5.7.4 Considerazioni conclusive ed estensioni

Possiamo concludere come l'attività di test dal punto di vista dinamico sia efficace e consenta di ridurre enormemente i tempi richiesti. Una volta preparate infatti le immagini di base, è sufficiente utilizzare lo script presentato poco sopra per creare delle macchine nei diversi tagli previsti dal provider e poi lanciare le attività di test, che permettono in breve tempo di ottenere risultati confrontabili.

Con l'evoluzione delle API di interoperabilità è inoltre possibile pensare a un supporto per l'importazione di immagini tra i diversi provider supportati, per cui sarà sufficiente crearla una volta e poi effettuare il deployment in maniera automatica presso i vari provider.

Per ciò che riguarda le possibili estensioni, l'attività di testing è, al momento, piuttosto limitata e adatta solo a numeri piuttosto piccoli; in previsione di carichi più elevati il primo elemento da introdurre sarebbe quello di avere più macchine dalle quali inviare le richieste, che eventualmente potrebbero essere smistate in base ad algoritmi più avanzati offerti dal load balancer utilizzato rispetto a quelli utilizzati nell'attività di test (ad esempio l'hash dell'indirizzo IP di provenienza).

Un altro possibile aspetto di interesse è quello di effettuare il test partendo da istanze su datacenter differenti: questo è un altro dei vantaggi dell'approccio cloud, in quanto, in una visione tradizionale, sarebbe necessario procurarsi una macchina da un fornitore locale per effettuare questo tipo di test, mentre ora è sufficiente creare un'istanza uguale alle altre ma in un datacenter differente, con chiari risparmi di tempo. Ciò ci può permettere di fare considerazioni interessanti sul servizio che è possibile offrire a utenti in diverse parti del mondo, soprattutto per quanto riguarda i tempi di risposta e la latenza.

Conclusioni

Il cloud computing è tra i temi più discussi e analizzati degli ultimi tempi all'interno del mondo dell'Information Technology. Se i vantaggi e i limiti iniziano ad essere chiari e ben definiti, dal punto di vista enterprise l'ostacolo maggiore, prima dell'adozione del cloud come di ogni altro nuovo paradigma, è la necessità di effettuare un'analisi delle soluzioni esistenti per massimizzare i vantaggi ottenibili. Tuttavia, il problema nella maggior parte dei casi, è il costo troppo alto richiesto da un'analisi approfondita, da cui l'esigenza di un processo approssimato, che possa tuttavia dare indicazioni importanti in modo rapido e a basso costo.

Questo è il tema centrale della tesi, che parte dall'idea di suddividere l'intero processo di migrazione in sei fasi principali: discovery, processamento, matching, spostamento, correzione delle configurazioni ed infine test. Queste fasi sono state suddivise in due categorie principali, una definita statica, in quanto basata su confronti tra dati raccolti sui sistemi in esame, l'altra dinamica, eseguita tramite test su sistemi in esecuzione; per ognuna di queste due parti si è cercato di sviluppare strumenti e procedimenti, che siano in grado di offrire un aiuto ai consulenti incaricati di analizzare le possibilità di migrazione di un ambiente preesistente. La parte, sulla quale ci siamo soffermati, è quella infrastrutturale, per la quale non ci sono molti esempi in letteratura; sono state invece tralasciate analisi approfondite sui problemi decisionali e organizzativi legati a una migrazione.

Lo strumento sviluppato per l'analisi statica è stato applicato a diversi casi reali, con risultati complessivamente positivi; essi sono stati infatti corretti e i tempi impiegati sono bassi, specialmente utilizzando un database locale. Le idee proposte e, in particolare, la flessibilità permessa dal modello relazionale dei dati, sono state apprezzate a livello aziendale, per cui si prevede un'integrazione e un utilizzo con soluzioni IBM precedenti.

La seconda parte, riguardante l'analisi che abbiamo classificato come dinamica, è necessariamente più generale; abbiamo comunque sviluppato un procedimento originale per

poter affrontare il problema in maniera adeguata. Le idee sviluppate sono state poi applicate ad uno scenario realistico, che ci ha permesso di fare considerazioni interessanti, specialmente per quanto riguarda il tema della scalabilità, importante all'aumentare dei dati da gestire. L'obiettivo di fornire un'idea basilare delle prestazioni delle configurazioni provate è stato raggiunto e, in conclusione, abbiamo anche introdotto alcune considerazioni sui costi, rese facili dalla modalità di pagamento a consumo.

Abbiamo, inoltre, affrontato il tema dell'interoperabilità tra i diversi provider cloud, proponendo uno script che permette di eseguire le operazioni di base previste nel nostro procedimento in maniera indipendente dai singoli provider. Tale script potrebbe in futuro essere esteso per gestire eventuali picchi al momento dell'esecuzione, che è una delle applicazioni più interessanti rese possibili dall'ambiente cloud.

Un ulteriore sviluppo potrebbe riguardare la maggior integrazione fra le varie fasi, pur mantenendo la loro indipendenza, che, a nostro giudizio, è anche uno dei punti di forza, perché consente di adeguarsi a un contesto molto vario. Ulteriori approfondimenti sarebbero necessari per quanto riguarda la fase di spostamento effettivo delle immagini, così da consentire una migrazione rapida piuttosto che per reinstallazione; tale parte è stata affrontata molto rapidamente in questa tesi e risulta, per ora, molto legata alle opzioni offerte dai singoli provider cloud.

Bibliografia

[AFG+10] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53:50–58, April 2010.

[AJP12] The Apache Jakarta Project. Jmeter. `http://jmeter.apache.org/`, 2012.

[Ama12] Amazon. Amazon elastic cloud computing. `http://aws.amazon.com/ec2/`, 2012.

[BB11] N. Borenstein and J. Blake. Cloud computing standards: Where's the beef? *Internet Computing, IEEE*, 15(3):74 –78, May-June 2011.

[BC11] Muhammad Ali Babar and Muhammad Aufeef Chauhan. A tale of migration to cloud computing for sharing experiences and observations. In *Proceeding of the 2nd international workshop on Software engineering for cloud computing*, SECLOUD '11, pages 50–56, New York, NY, USA, 2011. ACM.

[BH09] Luiz André Barroso and Urs Hölzl. *The Datacenter as a Computer - An Introduction to the Design of Warehouse-Scale Machines*. Morgan & Claypool, 2009.

[BPa10] IBM MI and IPR definition bridge between Gartner and IDC, 19th August 2010.

[CP12] The Cherokee Project. Cherokee web server. `http://www.cherokee-project.com/`, 2012.

[CS11] Yao Chen and Radu Sion. To cloud or not to cloud?: musings on costs and viability. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, SOCC '11, pages 29:1–29:7, New York, NY, USA, 2011. ACM.

[CUWS11] Emmanuel Cecchet, Veena Udayabhanu, Timothy Wood, and Prashant Shenoy. Benchlab: an open testbed for realistic benchmarking of web applications. In *Proceedings of the 2nd USENIX conference on Web application development*, WebApps'11, pages 4–4, Berkeley, CA, USA, 2011. USENIX Association.

[DMT10] DMTF. Open virtualization format specification 1.1.0. `http://www.dmtf.org/sites/default/files/standards/documents/DSP0243_1.1.0.pdf`, January 2010.

[DMT11] DMTF. Common information model 2.30.0. `http://www.dmtf.org/standards/cim`, September 2011.

[DP09] M. Dalheimer and F.-J. Pfreundt. Genlm: License management for grid and cloud computing environments. In *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, pages 132 –139, may 2009.

[DTB10] Amir Vahid Dastjerdi, Sayed Gholam Hassan Tabatabaei, and Rajkumar Buyya. An effective architecture for automated appliance management system applying ontology-based cloud discovery. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 104 –112, May 2010.

[FSW+10] R. Filepp, L. Shwartz, C. Ward, R.D. Kearney, K. Cheng, C.C. Young, and Y. Ghosheh. Image selection as a service for cloud computing environments. In *Service-Oriented Computing and Applications (SOCA), 2010 IEEE International Conference on*, pages 1 –8, Dec. 2010.

[GGL03] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. *SIGOPS Oper. Syst. Rev.*, 37:29–43, October 2003.

[GKHSS10] David Greenwood, Ali Khajeh-Hosseini, James Smith, and Ian Sommerville. The cloud adoption toolkit: Addressing the challenges of cloud adoption in enterprise. Technical Report arXiv:1003.3866, Mar 2010.

[Goo] Google. Google app engine. `https://appengine.google.com`.

[HSS+10] Mohammad Hajjat, Xin Sun, Yu-Wei Eric Sung, David Maltz, Sanjay Rao, Kunwadee Sripanidkulchai, and Mohit Tawarmalani. Cloudward bound: planning for beneficial migration of enterprise applications to the cloud. *SIGCOMM Comput. Commun. Rev.*, 41:243–254, August 2010.

[IBM] IBM. Smart cloud enterprise. `http://www.ibm.com/cloud-computing/us/en/`.

[IBM11] IBM. Cloud computing reference architecture 2.0. 2011.

[JMW+11] D. Jayasinghe, S. Malkowski, Qingyang Wang, J. Li, Pengcheng Xiong, and C. Pu. Variations in performance and scalability when migrating n-tier applications to different clouds. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 73 –80, July 2011.

[KGP+] Anil Kurmus, Moitrayee Gupta, Roman Pletka, Christian Cachin, and Robert Haas. A comparison of secure multi-tenancy architectures for filesystem storage clouds. Technical report, IBM Research Zurich.

[KHGS10] Ali Khajeh-Hosseini, David Greenwood, and Ian Sommerville. Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS. In *Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing*, CLOUD '10, pages 450–457, Washington, DC, USA, 2010. IEEE Computer Society.

[KHSBT] Ali Khajeh-Hosseini, Ian Sommerville, Jurgen Bogaerts, and Pradeep Teregowda. Decision support tools for cloud migration in the enterprise.

[KHSS10] Ali Khajeh-Hosseini, Ian Sommerville, and Ilango Sriram. Research challenges for enterprise cloud computing. *CoRR*, abs/1001.3257, 2010.

[LLM+08] Liang Liu, Ying Li, Qian Ma, Ke Wei Sun, Ying Chen, and Hao Wang. Automatic model-based service hosting environment migration. In *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pages 682 – 685, April 2008.

[LML+11] A. Lenk, M. Menzel, J. Lipsky, S. Tai, and P. Offermann. What are you paying for? performance benchmarking for infrastructure-as-a-service offerings. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 484 –491, July 2011.

[MDJV08] K. Magoutis, M. Devarakonda, N. Joukov, and N. G. Vogl. Galapagos: Model-driven discovery of end-to-end application-storage relationships in distributed systems. *IBM Journal of Research and Development*, 52(4.5):367 – 377, July 2008.

[Men02] D.A. Menasce. Tpc-w: a benchmark for e-commerce. *Internet Computing, IEEE*, 6(3):83 –87, May/Juny 2002.

[MG11] Peter Mell and Timothy Grance. The NIST definition of cloud computing. `http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf`, September 2011.

[Mic12] Microsoft. Microsoft Windows Azure. `http://www.microsoft.com/windowsazure`, 2012.

[MSB+07] H. Madduri, S. S. B. Shi, R. Baker, N. Ayachitula, L. Shwartz, M. Surendra, C. Corley, M. Benantar, and S. Patel. A configuration management database architecture in support of ibm service management. *IBM Systems Journal*, 46(3):441 – 457, 2007.

[Ope] OpenStack. *OpenStack Compute Developer Guide API 1.1*.

[Par66] Douglas Parkhill. *The challenge of the computer utility*. Addison-Wesley, 1966.

[Rai09] Geoffrey Raines. Cloud computing and SOA. October 2009.

[RUB] Rubis. `http://rubis.ow2.org/`.

[SB10] B. Speitkamp and M. Bichler. A mathematical programming approach for server consolidation problems in virtualized data centers. *Services Computing, IEEE Transactions on*, 3(4):266 –278, Oct.-Dec. 2010.

[SDQR10] Jörg Schad, Jens Dittrich, and Jorge-Arnulfo Quiané-Ruiz. Runtime measurements in the cloud: observing, analyzing, and reducing variance. *Proc. VLDB Endow.*, 3:460–471, September 2010.

[SS06] Eric Schmidt and Danny Sullivan. Conversation with Erick Schmidt hosted by Danny Sullivan. *Talk at Search Engine Strategies Conference*, 2006.

[SSR+10] Kunwadee Sripanidkulchai, Sambit Sahu, Yaoping Ruan, Anees Shaikh, and Chitra Dorai. Are clouds ready for large distributed applications? *SIGOPS Oper. Syst. Rev.*, 44:18–23, April 2010.

[SSS+05] Will Sobel, Shanti Subramanyam, Akara Sucharitakul, Jimmy Nguyen, Hubert Wong, Arthur Klepchukov, Sheetal Patil, Armando Fox, and David Patterson. Cloudstone: Multi-platform, multi-language benchmark and measurement tools for web 2.0. why we need new workloads. *Benchmarking*, 2005.

[Str10] Christof Strauch. NoSQL databases. pages 1–149, 2010.

[TKLF11] Van Tran, Jacky Keung, Anna Liu, and Alan Fekete. Application migration to cloud: a taxonomy of critical factors. In *Proceeding of the 2nd international workshop on Software engineering for cloud computing*, SECLOUD '11, pages 22–28, New York, NY, USA, 2011. ACM.

[TLF+11] V.T.K. Tran, K. Lee, A. Fekete, A. Liu, and J. Keung. Size estimation of cloud migration projects with cloud migration point (cmp). In *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*, pages 265 –274, Sept. 2011.

[Vog08] Werner Vogels. Beyond server consolidation. *Queue*, 6:20–26, January 2008.

[VRMCL08] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39:50–55, December 2008.

[WAB+10] C. Ward, N. Aravamudan, K. Bhattacharya, K. Cheng, R. Filepp, R. Kearney, B. Peterson, L. Shwartz, and C.C. Young. Workload migration into clouds. *Cloud Computing, IEEE International Conference on*, 0:164–171, 2010.

[WGR+09] Timothy Wood, Alexandre Gerber, K. K. Ramakrishnan, Prashant Shenoy, and Jacobus Van der Merwe. The case for enterprise-ready virtual private clouds. In *Proceedings of the 2009 conference on Hot topics in cloud computing*, HotCloud'09, Berkeley, CA, USA, 2009. USENIX Association.

[WSVY07] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. Black-box and gray-box strategies for virtual machine migration. In *Proceedings of the 4th USENIX conference on Networked systems design & implementation*, NSDI'07, pages 17–17, Berkeley, CA, USA, 2007. USENIX Association.

[YBDS08] L. Youseff, M. Butrico, and D. Da Silva. Toward a unified ontology of cloud computing. In *Grid Computing Environments Workshop, 2008. GCE '08*, pages 1 –10, Nov. 2008.

[ZCB10] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1:7–18, 2010. 10.1007/s13174-010-0007-6.

[ZL] Gong Zhang and Ling Liu. Why do migrations fail and what can we do about it?

[ZRE+11] Rui Zhang, R. Routray, D.M. Eyers, D. Chambliss, P. Sarkar, D. Willcocks, and P. Pietzuch. IO Tetris: Deep Storage Consolidation for the Cloud via Fine-Grained Workload Analysis. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 700 –707, July 2011.

Ringraziamenti

In certi momenti ho dubitato di riuscire ad arrivare a scrivere queste poche sudate righe e le ore di dubbio, ansia, paura non sono state poche, come le volte in cui ho pensato di mollare tutto. Alla fine invece ce l'ho fatta e la soddisfazione è enorme, ma molto lo devo a tutti quelli che mi hanno aiutato e che cercherò di ricordare qua sotto, scusandomi in anticipo con tutti quelli che dimenticherò. Per questo il primo grosso ringraziamento va sicuramente a mio padre e mia madre, che mi hanno sempre sostenuto e spinto ad andare avanti nei momenti di difficoltà (certe volte anche stressandomi!). Graziis di cûr anche a mia sorella Michela per le cene, i consigli, i lavaggi, la compagnia e a David che è sempre stato disponibile.

Grazie alla mia ragazza Martina, una persona veramente speciale che ha illuminato la maggior parte di questa laurea specialistica e che mi ha sempre incoraggiato e sopportato nei miei (frequenti) momenti pessimistici.

Grazie poi a Luca perché è un amico raro, sempre disponibile, oltre che mago dell'SQL e un po' di tutto. Grazie a Stefano, Giacomo, Valeria, insostituibili coinquilini e sempre super disponibili. Grazie a Giuseppe, questa tesi è partita anche dal Super Liquidator di Grado. Grazie di tutto a Jacopo, che ha reso i lavori di gruppo un po' meno pesanti. Grazie anche a Nicola, Marco con il bingo, Ugo, Michele, Piero e tutti gli altri per la compagnia.

Grazie poi a Enrico per le serate a Milano, e grazie a tutta la gente che mi ha fatto compagnia nell'avventura milanese, Luca, Lorenzo, Alberto e Giuseppe, con cui ho lavorato e mi son divertito tra una pausa alle macchinette ai piani alti e un salto giù al Forum. Specialmente a Luca a cui ho scroccato un letto in tempi di difficoltà. Grazie poi a Paolo e Svetlana, che sono stati splendidi coinquilini.

Infine vorrei ringraziare il prof. Bellavista per avermi proposto lo stage da cui è nata questa tesi e avermi quindi dato l'opportunità di esplorare un tema di grande interesse come il cloud. Grazie poi a tutte le persone di IBM che mi hanno aiutato in questi mesi,

in particolare Emilio Lucotti, sempre pronto a darmi idee, suggerimenti, documenti utili e a rivedere la mia tesi, che senza il suo aiuto sarebbe venuta molto peggio, e Mariano Ammirabile, che mi ha insegnato molte cose.