Alma Mater Studiorum · Università di Bologna

facoltà di scienze matematiche, fisiche e naturali
Corso di Laurea Triennale in Informatica

Garanzie del servizio in ambienti di cloud computing: uno studio sperimentale

Tesi di Laurea in Reti di Calcolatori

Relatore:
Chiar.mo Prof.
Fabio Panzieri

Presentata da:
Emilio Junior Francischetti

Sessione I
Anno Accademico 2010/2011

Ai miei genitori.

Indice

1 Introduzione **4**

2 Stato dell'arte **13**
2.1 Architettura di un data center 13
2.2 Network 14
2.3 Raffreddamento 15
2.4 Alimentazione 16
2.5 Data center classici 16
2.6 Data center modulari 18
2.7 Virtualizzazione 19
2.8 File system distribuiti 20
2.8.1 Google File System 20
2.9 Load Balancing 22
2.9.1 Google MapReduce 22
2.10 Database management system distribuiti 23
2.10.1 Google BigTable 23
2.11 Cloud Framework 24
2.11.1 Eucalyptus 24
2.12 Sfide alla ricerca 27
2.12.1 Allocazione automatica 27
2.12.2 Migrazione macchine virtuali 28
2.12.3 Consolidamento server 28
2.12.4 Gestione energetica 28
2.12.5 Gestione e analisi del traffico di rete 29
2.12.6 Sicurezza dei dati 29
2.12.7 Piattaforme software 29
2.12.8 Tecnologie di archiviazione dati 30
2.12.9 Nuove architetture di cloud 30

3 Service Level Agreement **31**
3.1 Cos'è un contratto di SLA 31
3.2 SLA Framework 31
3.3 WSLA Framework 31
3.3.1 Obiettivi del progetto 32
3.3.2 Le fasi di WSLA 33
3.3.3 Il linguaggio WSLA 35
3.3.4 Implementazioni di WSLA Framework 36
3.4 WS-Agreement Framework 36
3.4.1 Struttura di Accordi e Template 37
3.4.2 Modello di Interazione fra Accordi 38
3.4.3 Cremona: Architettura di WS-Agreement 39
3.5 Confronto tra WSLA e WS-Agreement 41

4 Soluzioni esistenti **43**
4.1 Google AppEngine 43
4.2 Microsoft Azure 47
4.3 Amazon Elastic Compute cloud 51

5 Sperimentazione **54**
5.1 Scenario 54
5.2 Primo test 55
5.3 Secondo test 57
5.4 Terzo test 59
5.5 Considerazioni 61

6 Conclusioni **62**

Capitolo 1

Introduzione

La definizione di cloud computing può variare dal punto di vista di chi la definisce. Una delle più generiche è quella data dal National Institute of Standards and Technology (NIST):

> Il cloud computing è un modello abilitante un accesso comodo ed on-demand ad un pool condiviso di risorse di calcolo configurabili che possono essere velocemente ottenute e rilasciate con minimo sforzo di gestione ed una limitata interazione con il fornitore di servizi.

Il termine cloud computing fu coniato nel 1997 da Ramnath K. Chellappa [1]. Questa invenzione non si basa su nulla di nuovo, ma unisce tecnologie e concetti già presenti da decenni, come: utility computing, grid computing e virtualizzazione.

- **Utility computing:** nel 1961, John McCarthy [2] fu il primo a dire (in un discorso fatto durante le celebrazioni del centenario del MIT[1]) che il metodo time-sharing (condivisione a tempo) dei computer può condurre verso un futuro dove la potenza dei calcolatori ed anche specifiche applicazioni possono essere vendute secondo il modello economico dell'utilità (come succede per l'acqua ed elettricità). Quest'idea era molto popolare alla fine degli anni 60, ma scomparì intorno alla metà degli anni 70, quando divenne chiaro che l'hardware, il software e le telecomunicazioni del tempo non erano pronte.

- **Grid computing:** come spiegato in [3] il grid computing nasce nella seconda metà degli anni 90, è consiste nell'unione di risorse di calcolo, distribuite geograficamente e collegate tramite internet. Tali risorse appartenenti a diverse organizzazioni, sono utilizzate per elaborazioni fortemente data-intensive in modo da ridurre i tempi di calcolo.

[1] Massachusetts Institute of Technology http://web.mit.edu/

- **Virtualizzazione:** alla fine degli anni 60, emerse che l'utilizzo effettivo di un server si aggirava tra il 15% e il 20% della sua capacità. Si decise quindi di consolidare in un unico mainframe un gran numero di server, in modo da diminuire i costi e massimizzare il loro utilizzo. Questa tecnologia consiste nel suddividere le risorse della macchina fisica, creando una macchina virtuale con un proprio sistema operativo, che compete con i sistemi operativi delle altre macchine virtuali per l'esecuzione delle proprie operazioni da parte della macchina fisica.

Il cloud computing si differenzia in tre categorie in base al servizio offerto:

- **Software as a Service (SaaS):** consiste nel utilizzo di un applicazione su un server remoto, liberando l'utente dai problemi di installazione e fornendo i servizi tipici di un data center (e.s RAID[2] Storage). Alcuni esempi esistenti sono Google Docs[3] ed Office 360[4].

- **Platform as a Service (Paas):** fornisce ai Service provider un ambiente di sviluppo (tools, librerie, database ecc) per interagire con l'infrastruttura della cloud, nel creare le proprie applicazioni. Gli attori principali di questa tipologia sono Google AppEngine[5] (Java e Pyhon) e Microsoft Azure[6] (.NET e altri).

- **Infrastructure as a Service (IaaS):** permette l'utilizzo di risorse in remoto dando libera scelta su come configurare le macchine virtuali utilizzate. Gli attori principali sono Amazon Elastic Compute Cloud[7] (EC2), GoGrid[8] e FlexiScale[9].

  La figura 1.1 mostra i diversi gradi di gestione del servizio. Nel primo caso, un utente di una SaaS non può modificare nulla al di fuori delle impostazioni permesse dal servizio stesso. L'utente si limita al solo utilizzo secondo i termini del contratto di utilizzo. Nel secondo caso, l'utente di una PaaS ha piena libertà (nei limiti della legge) sulla creazione della sua applicazione, ma vincolata all'utilizzo della piattaforma della PaaS. Per esempio con Google AppEngine posso creare qualsiasi applicazione, ma devo per forza usare il loro database non relazionale e un linguaggio di sviluppo tra Java e Python. Nel terzo caso quello di un utente di una IaaS, troviamo il grado massimo di gestione. É possibile utilizzare qualsiasi tools, database, configurazione di rete e politica di sicurezza che si desidera, proprio come

[2]Redundant Array of Independent Disks è un sistema informatico che usa un insieme di dischi rigidi per condividere o replicare le informazioni

[3]docs.google.com

[4]http://www.office3sixty.com/

[5]http://code.google.com/intl/it-IT/appengine/

[6]http://www.microsoft.com/windowsazure/

[7]http://aws.amazon.com/ec2/

[8]http://www.gogrid.com/

[9]http://www.flexiant.com/products/flexiscale/

se le macchine fossero nostre. L'unica limitazione è sulla gestione delle macchine fisiche.

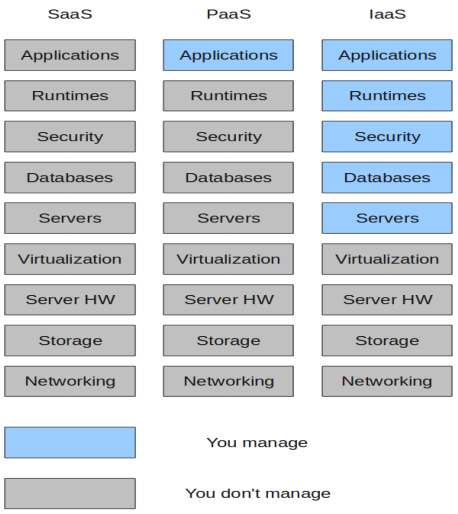


Figura 1.1: Grado di gestione dei servizi tratta da [4]

Un altra categorizzazione avviene in base al metodo di accesso all'infrastruttura di Claud:

- **Public cloud:** è un infrastruttura accessibile a tutti, dove il cloud provider si accolla l'investimento iniziale di capitali per costruire i data center, le spese di manutenzione/rinnovo e gestione dell'infrastruttura.
- **Private cloud:** conosciuto anche come internal cloud, consiste un infrastruttura privata, dedicata alle esigenze di una singola azienda. Offre il massimo controllo sulle macchine, soprattutto a livello di sicurezza dei dati.
- **Hybrid cloud:** unisce i benefici di possedere risorse quasi illimitate (Public cloud) alla possibilità di massimo controllo sulle macchine (Private cloud). Un possibile scenario di realizzazione è una Private cloud con mansioni di Database, con tecnici interni che garantiscono la sicurezza dei dati aziendali, affiancata ad una parte puramente di calcolo offerta da una Public cloud.
- **Virtual Private cloud (VPC):** un alternativa per superare le limitazioni delle Public e Private cloud. Consiste in una rete VPN (Virtual Private Network) sicura, pienamente configurabile (regole Firewall, ecc) che poggia su Public cloud.

Come mostra la figura 1.2, gli attori coinvolti nel mondo del cloud computing sono tre:

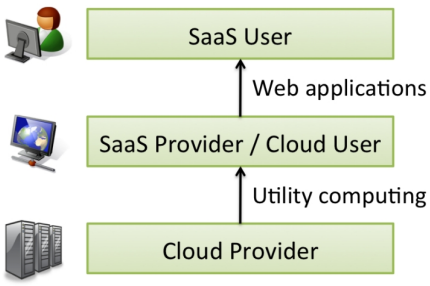


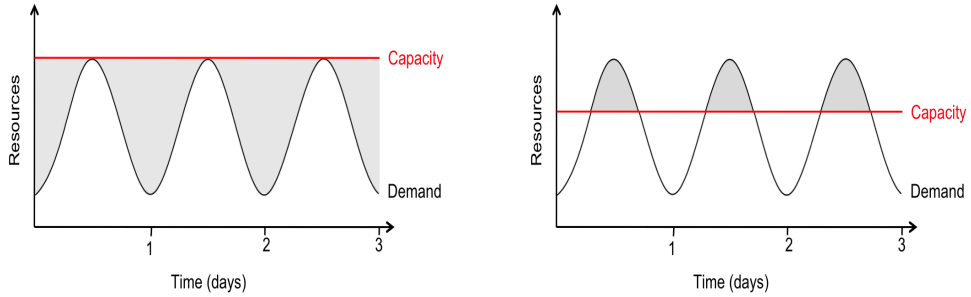
Figura 1.2: Attori cloud computing tratta da [5]

- **Saas user:** gli utenti possono accedere ai servizi dei SaaS provider "anytime, anywhere"(in ogni momento, in ogni luogo). Dal momento che le applicazioni girano su un Browser, gli utenti possono accedervi da qualsiasi dispositivo che lo permette.
- **SaaS provider:** trovano nei cloud provider un modo rapido ed economico per distribuire i loro servizi al largo pubblico, potendosi permettere potenza di calcolo che senza grossi fondi non avrebbero mai potuto avere.
- **Cloud provider:** si fa carico di tutte le problematiche relative all'infrastruttura fornendola come servizio.

A livello monetario il cloud computing offre svariate riduzioni di costi a tutti gli attori del cloud computing:

- **SaaS user:** gli utenti con un minimo costo usufruiscono dell'infrastruttura di un data center. Un tipico esempio è DropBox[10] che permette con pochi dollari mensili, di godere delle capacità di archiviazione dati (RAID) e sicurezza tipiche di un data center.
- **SaaS provider:** con il sistema di pagamento pay as you go, pagano posticipatamente solamente le risorse effettivamente consumate, eliminando i rischi di over-provisioning in figura 1.3 (a) (pagamento di server poco utilizzati ma necessari per far fronte a sporadici picchi di richieste) e underprovisioning in figura 1.3 (b) (potenza di calcolo insufficiente alle richieste con conseguente perdita di clienti per disservizi). Eliminando anche la necessità di un capitale iniziale per costruire l'infrastruttura di calcolo. La tipica startup californiana non dove più rivolgersi

[10]http://www.dropbox.com/



(a) Overprovisioning (b) Underprovisioning

Figura 1.3: Utilizzo risorse tratto da [5]

alle joint venture per reperire fondi, cadendo nella trappola dell'ingresso forzato in borsa per una remunerazione economica da parte della join venture.

- **Cloud provider:** grazie all'enorme quantità di risorse, vengono a crearsi economie di scala riducendo il costo dei servizi rispetto ad un data center di media dimensione. L'hardware comprato in grosse quantità, permette di ottenere significativi sconti da parte dei venditori. Ma il costo maggiore per le imprese IT, è l'energia eletrica. Grazie alla dislocazione dei data center in diversi stati (per avere multi-latency e diminuzione dei rischi di catastrofi naturali), i cloud provider riescono a scegliere le zone con costo energetico minore ottenendo contratti a volume, con sconti anche del 15% sul prezzo base.

In [5] vengono elencati alcuni ostacoli allo sviluppo del cloud computing.

- **Disponibilità del servizio:** la disponibilità del servizio è un fattore importantissimo per i SaaS provider, che possono subire notevoli danni d'immagine dovuti a disservizi subiti dagli utenti. Nonostante un cloud provider utilizzi svariati data center, sistemi di replicazione (multi Network provider, RAID) e notifica dei guasti (monitor di macchine virtuali); usando gli stessi software in tutta l'infrastruttura si può creare un "single point of failure", nel caso questi software presentino bug. La seguente tabella elenca alcuni casi accaduti nei recenti anni.

Servizio	Causa interruzione	Durata	Data
Amazone S3	sovraccarico autenticazione	2 ore	15/02/08
Amazone S3	errore di un bit nel protocollo gossip	6-8 ore	20/07/08
AppEngine	errore di programmazione	5 ore	17/06/08
Gmail	errore sistema contatti	1.5 ore	11/08/08

I SaaS provider potrebbero perdere clientela a causa dei disservizi subiti dai loro clienti, mentre potenziali SaaS provider potrebbero essere scoraggiati a usufruire

dei servizi di un cloud provider che soffre di mancanza di continuità di servizio. Un altro ostacolo alla continuità di servizio è la minaccia di un attacco **Distributed Denial of Service (DDoS)**. Questo attacco consiste nel sovraccaricare la rete bersagliata con richieste di risorse, lasciando la vittima impotente a gestire le richieste legittime. Le motivazioni dietro questi attacchi posso essere di tipo politico, personale ma soprattutto a fini estorsivi. I malintenzionati chiedono alle vittime dai 10.000$ ai 50.000$ per non rendere irraggiungibili le loro applicazioni, con conseguente perdità di immagine e clientela. Queste attacchi necessitano di molte risorse, e vengono eseguiti grazie al noleggio di **botnet**. Una botnet consiste in un gruppo di macchine infettate che si mettono in ascolto su canali irc[11] , in attesa di ordini da parte degli affittuari. Queste macchine vengono vendute in gruppi di 1000 a circa 0,03$ per macchina alla settima. Grazie alla velocità dell'autoscale il cloud offre una valida difesa a queste tipologie di attacchi. Supponiamo che un attacco lanciato con 500.000 bots generi 1 GB/secondo di traffico di rete. Il costo del reperimento delle macchine per l'attacco è di 15.000$. Secondo le attuali tariffe l'attacco dovrebbe durare più di 30 ore per costare alla vittima più della cifra estorta. Dal punto di vista teorico l'autoscale si dimostra un forte deterrente al DDos, ma recenti fatti di cronaca bocciano questa tesi.

Un cliente di Amazon EC2 colpito duramente da DDoS

> Una cattiva pubblicità per Amazon EC2, il noto servizio di cloud computing erogato dal gigante dell'e-commerce. Durante la fine della scorsa settimana un cliente del servizio, la compagnia Bitbucket[12], fornitrice di un servizio di code hosting, ha riscontrato alcuni problemi nel servizio. Nel fine settimana l'istanza e il relativo EBS del sito, completamento ospitato all'interno del servizio di Amazon, sono risultati down per oltre 19 ore consecutive, con gravi perdite economiche e di immagine per la compagnia. Il problema è dovuto ad un massiccio attacco DDoS che ha messo in ginocchio il servizio di Amazon, lasciando che lo staff non riuscisse a porvi rimedio nel giro di poche ore dopo essersi accorto delle problematiche. Un comunicato ufficiale di Amazon parla di un attacco mirato al sito dell'utente, lo staff ha cercato di lavorare con i proprietari per mitigare l'attacco e far tornare online il sito non appena possibile. Lo sconforto di Bitbucket, che ha voluto testimoniare in dettaglio quanto accaduto sul blog della compagnia, sembra essere dovuto principalmente alla scarsa comunicazione ricevuta da Amazon in un primo momento,

[11] Internet Relay Chat (IRC) è stata la prima forma di comunicazione istantanea (chat) su Internet. Consente sia la comunicazione diretta fra due utenti che il dialogo contemporaneo di interi gruppi di persone in stanze di discussione chiamate canali.

[12] https://bitbucket.org/

oltre alla mancanza di soluzioni per una infrastruttura che non dovrebbe consentire che simili attacchi blocchino il servizio per oltre 10 ore. Bitbucket parla anche di numerose comunicazioni con Amazon, divenute risolutive solo nel momento in cui è stato acquistato il servizio di assistenza di livello gold. Il problema era dovuto all'impossibilità di lettura dei dati nell'EBS, lo storage usato dalle istanze EC2, per via dell'attacco DDoS. A poco è servito trasferire l'istanza in un nuovo data center, come testimonia il post, e solo dopo diverse ore di lavoro dei network engineers Amazon il problema è stato risolto. Un brutto episodio per Amazon e per i servizi di cloud computing, ma di fatto un problema che pone dei limiti non solo al servizio ma all'approccio di Bitbucket, come fa notare The Register[13], **lo sbaglio sta infatti nel pensare che la collocazione presso un solo provider del sito sia una soluzione per rimanere sempre e comunque online**, anche se si tratta di cloud computing e di una infrastruttura che dovrebbe comunque poter dare una migliore risoluzione a questi problemi.

08/10/2009 di Stefano Bellasio su http://www.hostingtalk.it

- **Blocco dei dati:** Grazie all'utilizzo di linguaggi come Java e Python (fortemente multipiattaforma) sarebbe molto facile spostare la propria applicazione da un cloud provider all'altro. Ma le Application Programming Interface (API) per il trattamento dei dati restano fortemente proprietarie, rendendo praticamente impossibile lo spostamento dell'applicazione. Inoltre, nel caso di fallimento o scarsa gestione del cloud provider i SaaS provider e i SaaS user possono perdere i loro dati, come avvenne al 45% degli utenti di The Linkup (storage online simile a Dropbox). Per risolvere questi inconvenienti si può cercare di standardizzare le tecnologie di archiviazione in modo che possano collaborare tra loro.

- **Privacy dei dati:** Molte aziende sono riluttanti a immagazzinare i propri dati sensibili nei cloud provider per motivi di privacy. Per esempio in America grazie all'USA PATRIOT Act (legge antiterrorismo), il governo può ordinare al cloud provider, di accedere ai dati di una azienda sospetta di terrorismo. I cloud provider offrono la possibilità di salvare i propri dati in stati in cui non vi siano leggi simili.

[13] http://www.theregister.co.uk/

Uno dei maggiori detrattori del cloud computing è Richard Stallman che afferma in un intervista al giornale The Guardian:

> É stupido. Peggio della stupidità. É una campagna di marketing. Qualcuno dice sia inevitabile. Ogni volta che sento frasi del genere, suonano come una campagna di bussines per renderlo vero. Gli utenti devono tenere il controllo dei propri dati, e non cederlo a terze parti.

Una soluzione al mantenimento della privacy degli utenti è la possibilità di salvare i propri dati in maniera criptata, soluzione però non applicabile ai database per motivi di consistente perdita di performance.

- **Trasferimento dei dati:** dal momento che le applicazioni stanno diventando sempre più "data-intensive" il costo del trasferimento dati pesa fortemente sui bilanci aziendali. Una soluzione è di spedire fisicamente grandi quantità di dati su dischi. Per esempio, vogliamo spedire 10 TB da U.C. Berkeley ad Amazon. Con una media di 20 Mbit/sec di Upload ci impiegheremmo più di un mese, con una spesa di circa 1000$. Se invece salvassimo i dati in 10 dischi da un TB, e li spedissimo tramite corriere avremmo i nostri dati a destinazioni in 1 giorno al costo di 400$.

- **Imprevedibilità delle performance:** la condivisione di CPU e I/O da parte delle macchine virtuali può portare a interferenze sulla valutazione delle reali capacità di calcolo possibili. Per ovviare a questi inconvenienti si può migliorare la gestione degl'interrups da parte dei sistemi operativi e sostituire gli Hard Disk meccanici con Solid State Disck (SSD) che migliorano di gran lunga le performance in lettura e scrittura.

Capitolo 2

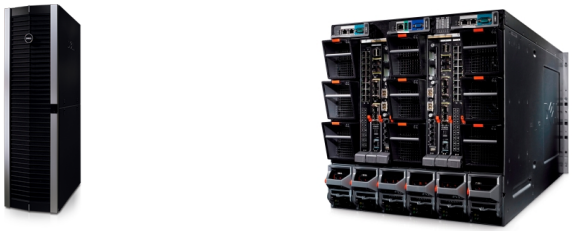
Stato dell'arte

In questo capitolo analizzeremo le tecnologie Hardware e Software utilizzate nel cloud computing e le sfide attuali nell'ambito della ricerca .

2.1 Architettura di un data center

Il data center è l'elemento centrale del cloud computing, contiene migliaia di dispositivi (servers, routers, switches ecc). Le macchine che compongono il cluster sono server con risorse omogenee, con hardware ottimizzato al tal fine, come i Server PowerEdge di Dell[1] o i ProLiant di HP[2]. Questi computer sono dotati di schede madri con più socket, per ospitare anche 8 processori multicore su una singola scheda madre. I processori in questione presentano istruzioni ottimizzate per l'utilizzo in ambiente server. Tipici processori sono gli intel[3] xeon e gli amd[4] opteron. Anche i case, che contengono i componenti assumono forme ottimizzate per diminuire lo spazio occupato. Tipiche soluzioni sono i server rack in figura 2.1 (a) che permettono la collocazione verticale in appositi armadi che ospitano dai 12 ai 48 dispositivi (server e dispositivi ausiliari) e la nuova tendenza nel mondo dell'IT (Information Technology) i server blade in figura 2.1 (b) che incorporano servizi di raffreddamento, connettività e alimentazione negli armadi blade, simili agli armadi rack ma con i servizi precedentemente elencati.

[1] http://www.dell.com/
[2] http://www8.hp.com/
[3] http://www.intel.com/
[4] http://www.amd.com/



(a) armadio rack (b) armadio blade

Figura 2.1: Tipologie di server tratte dal sito http://www.dell.com/

2.2 Network

Come spiegato in [6], l'infrastruttura di rete è una parte molto critica, essendo i data center ambienti di calcolo distribuito. Necessità quindi di sistemi di replicazione per essere tollerante ai guasti. Come mostra la figura 2.2 la rete di un data center si suddivide in 3 parti:

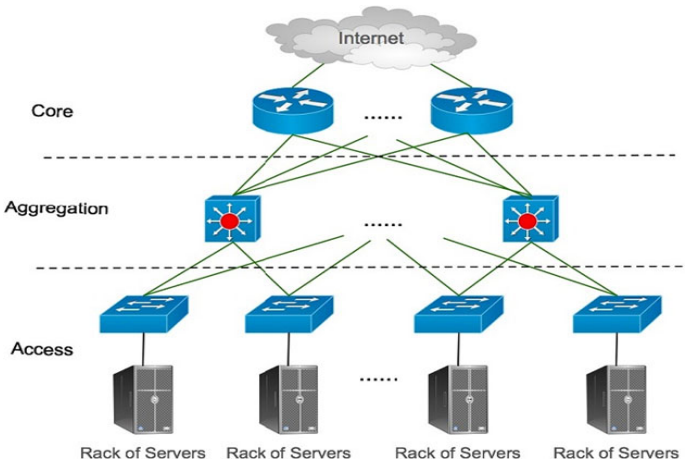


Figura 2.2: Infrastruttura di rete di un data center tratta da [6]

- **Acces Layer:** ogni server e connesso alla rete attraverso un cavo da 1 Gbps collegato ad uno Switch connesso con l'Aggregation Layer.
- **Aggregation Layer:** fornisce la connettività interna al data center, fornendo importanti funzioni come servizi di dominio, locazione e distribuzione del carico operativo. In questo livello sono presenti dispositivi ripetuti per rendere la rete tollerante i guasti.

- **Core:** rappresenta la porta sul mondo esterno, qui si concentra la sicurezza del data center. Vengono utilizzati molteplici Internet provider per garantire la continuità di connessione.

Questa infrastruttura di rete permette di godere delle seguenti proprietà:

- **Uniforme capacità di traffico:** grazie alla ripetizione di switch, si ottiene una potenza di trasferimento maggiore di quella richiesta, limitando cosi la velocità di trasferimento solo alla capacità delle schede di rete dei singoli server. Un altro vantaggio è che la comunicazione tra i server all'interno del data center, gode della potenza di una rete LAN dando la possibilità di usare la migrazione di macchine virtuali per rimediare a guasti o per politiche di distribuzione di carico operativo.
- **Elasticità e Scalabilità:** grazie alla ridondanza di componenti la rottura di un cavo di rete con uno switch non blocca la comunicazione, dal momento dell'esistenza di percorsi di routing alternativi garantisce la continiutà di comunicazione. Inoltre l'aggiunta di nuovi server è molto semplice e non richiede modifiche strutturali.

2.3 Raffreddamento

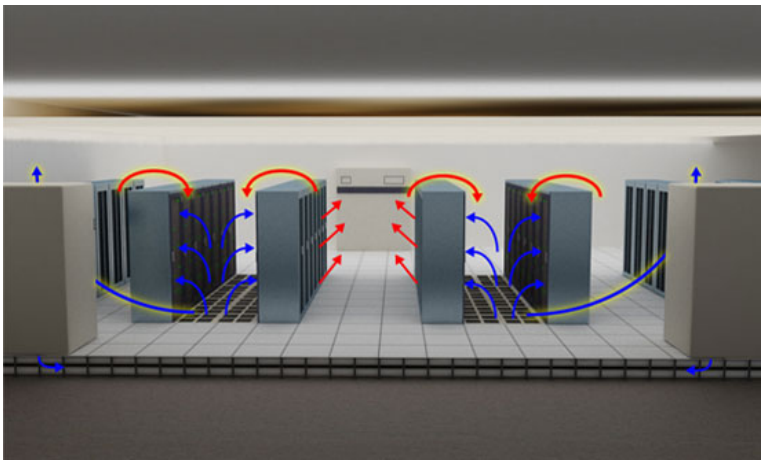


Figura 2.3: Sistema di raffreddamento CRAC tratta dal sito http://www.42u.com/

Dato l'utilizzo continuo di apparecchiature elettrice il problema del surriscaldamento della macchine e molto importante. Le soluzioni al tal problema sono varie, una della più usate e il sistema CRAC (Computer Air Room Conditioning). Come mostra la figura 2.3, grazie ad un piano rialzato con una grata forata è possibile far salire l'aria fredda (grazie ad un aumento di pressione) nella sala server, che entra nel rack tramite aperture frontali, raffreddando i componenti ed uscendo dal retro del rack. Vengono a

crearsi corridoi di aria calda e fredda che in passato venivano murati per migliorare la dispersione di temperatura.

2.4 Alimentazione

L'infrastruttura energetica di una data center è composta principalmente da tre elementi:

- **Generatore elettrico:** dispositivo destinato a produrre energia elettrica a partire da una diversa forma di energia, solitamente combustibili fossili. Serve a garantire la potenza energetica necessaria in caso si verifichino guasti alle fonti principali di energia. Grazie ai grandi serbatoi di combustibile può garantire energia elettrica anche per settimane.
- **Gruppi di continuità:** in inglese Uninterruptible Power Supply (UPS), è un'apparecchiatura che si usa per mantenere costantemente alimentati elettricamente in corrente alternata apparecchi elettrici. Si rivela necessario laddove le apparecchiature elettriche non possono in nessun caso rimanere senza corrente. È utilissimo soprattutto nei paesi dove si producono frequenti e sistematici black-out.
- **Unità di distribuzione:** dall'inglese Power Distribution Units (PDU), sono dispositivi collocati all'interno degli armadi rack che forniscono la giusta potenza eletrica ai vari componenti presenti.

2.5 Data center classici

In questo paragrafo analizzeremo un data center classico, nello specifico analizzeremo la webfarm italiana Aruba[5], fornendo dati tratti dal suo sito.

- **Connettività:** La connessione alla Big Internet è fondamentale per il funzionamento della Webfarm. Per questo sono state realizzate connessioni multiple con diversi fornitori, utilizzando le migliori tecnologie disponibili. Ogni connessione ha un punto di ingresso nell'edificio ed un percorso geografico diverso rispetto alle altre, in modo da ridurre drasticamente le possibilità di guasti simultanei. Inoltre la combinazione di fornitori italiani ed esteri oltre alla connessione diretta al principale punto di scambio italiano garantiscono non solo ridondanza ma anche le massime prestazioni possibili.
  - 5Gbit/s con Telecom Italia tramite 2 link Packet-over-Sonet da 2,5Gbit/s
  - 20Gbit/s con Wind tramite 2 link 10GigaEthernet da 10Gbit/s

[5]http://www.aruba.it/

  - 2Gbit/s con il Mix di Milano tramite 2 link GigaEthernet da 1Gbit/s
  - 20Gbit/s con Cogent Communications tramite 2 link 10GigaEthernet da 10Gbit/s
    In totale 47Gbit/s di connettività ridondata e gestita dinamicamente tramite protocollo BGP4 (AS31034).
  - 2 Router Cisco classe 12.000 in bilanciamento per le connessioni esterne
  - 2 Router Cisco classe 7.600 in bilanciamento per le connessioni interne
  - 2 Switch Cisco Catalyst 6513 con oltre 500 porte Gigabit per la distribuzione a tutti gli armadi rack.
  - Oltre 500 Switch per la distribuzione locale in ogni singolo rack. Cablaggio LAN interna interamente certificato categoria 6

- **Alimentazione:** Altro elemento fondamentale per il perfetto funzionamento della struttura è l'energia elettrica. Per ottenere la massima qualità e protezione, abbiamo realizzato una nostra stazione di trasformazione da 5 Megawatt connessa ad Enel tramite linee ad alta tensione. Questa stazione è connessa al Power Center interno dove sono presenti 5 UPS da 500Kva in parallelo ridondato ed i relativi gruppi di batterie che garantiscono la presenza immediata di alimentazione elettrica d'emergenza e l'isolamento dalla rete elettrica esterna. In caso di interruzione dell'energia da parte di Enel, i due generatori diesel intervengono entro 60 secondi ed hanno autonomia di diversi giorni anche in assenza di rifornimenti. Infine la rete di distribuzione interna fornisce 4 diverse linee di alimentazione ad ogni armadio rack.

- **Climatizzazione:** Il sistema di condizionamento garantisce temperatura ed umidità costanti all'interno delle sale dati. La potenza complessiva degli impianti è tale da garantire un adeguato raffreddamento anche in caso di guasto di un terzo delle unità installate. É inoltre presente un impianto per il ricambio dell'aria che tramite espulsione forzata verso l'esterno consente un ricambio completo nelle sale dati in 2 ore.

2.6 Data center modulari

Figura 2.4: BlackBox di Sun Microsistem tratta dal sito http://www.sun.com/

Negli ultimi anni è stato abbandonato il concetto classico di data center, creando il concetto di data center modulari (MDC) in figura 2.4. Il MDC è un datacenter mobile costruito all'interno di un container standar da 20 piedi (6 metri circa). Per il suo funzionamento necessita solo di una attacco alla rete elettrica, una connessione ad internet ed un impianto idrico (acqua fredda in entrata, calda in uscita). Ovviamente il trasporto di queste soluzioni, considerando la presenza al loro interno di server, prevede un procedimento delicato, e rende quindi il trasporto una possibilità solamente in casi di vera necessità. Un datacenter con più di 280 server può essere velocemente consegnato dentro un normale container e connesso all'infrastruttura, in posti che normalmente non potrebbero ospitare una struttura tradizionale.

2.7 Virtualizzazione

Come spiegato in [7], la virtualizzazione è una tecnologia che permette di partizionare un computer in parti completamente isolate, ognuna con il proprio ambiente software come se fosse un vero computer. Questo ambiente è chiamato virtual macchine (VM). Ogni VM presume di avere a disposizione l'intero pool di risorse fisiche, ma non è così. L'interazione con l'hardware passa per il virtual macchine monitor (VMM) detto anche hypervisor, che controlla la corretta condivisione delle risorse fisiche. L'VMM ha interfacce di basso livello per virtualizzare l'hardware sottostante, ma resta sempre un programma che gira su un sistema operativo privilegiato, chiamato sistema operativo host o domain0 che permette il funzionamento delle risorse. La situazione è mostrata in figura 2.5.

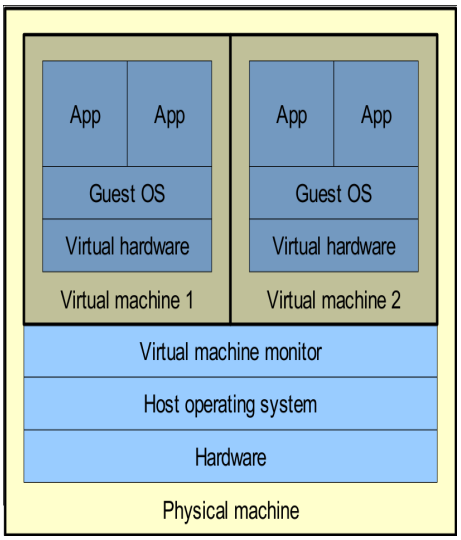


Figura 2.5: Struttura di una macchina fisica virtualizzata tratta da [7]

Le VM offrono svariati vantaggi:

- **Server:** permette l'esecuzione isolata di applicazioni su una singolo computer, utilizzando le macchine molto più efficientemente, riducendo i costi di hardware.
- **Sviluppo software:** la velocità con cui si puo installare un sistema operativo su una macchina virtuale (senza formattare la macchina fisica) permette il testing e debug di software multipiattaforma.
- **Software instabile:** grazie al completo isolamento delle VM possiamo utilizzare software che presentano bug o instabilità, senza pregiudicare la sicurezza dei dati che risiedono su altre VM.

- **Honeypots:** grazie ad una VM accedibile da internet possiamo creare una trappola inserendo in essa finti dati sensibili, la quale registrerà informazioni sugli attacchi subiti, che serviranno poi ai tecnici per mettere in sicurezza le macchine con i veri dati sensibili.

2.8 File system distribuiti

In un Cluster non è possibilie salvare i dati nei file system locali alle singole macchine perchè non sarebbero accessibili al resto delle macchine del cloud e comporterebbe problemi di perdità dei dati a causa di guasti. Nasce quindi l'esigenza di un file system distribuito.

2.8.1 Google File System

Come spiegato in [8], il google file system (GFS), racchiude gli scopi principali dei tipici file system distribuiti (performance, scalabilità e disponibilità), con le esigenze di tasferimento e storage all'interno dei cluster di google. I progettisti hanno esaminato le soluzioni esistenti estraendo i seguenti punti chiave:

- i guasti dei componenti sono la norma piuttosto che l'eccezione. I cluster sono composti da milioni di macchine assemblate con hardware economico, che non garantisce sicurezza di storage. Anche errori umani, di rete o bug possono minare la persistenza dei dati. Quindi è necessario un sistema integrato di monitoraggio di errori e ripristino dei dati per garantire la tolleranza ai guasti.

- Le principali operazioni di lettura consistono in grandi flussi di dati o piccole porzioni random all'interno del file.

- Le principali operazioni di scrittura consistono in aggiunte di dati alla fine del file (append mode) o piccole scritture random all'interno del file.

Architettura

Un GFS cluster è composto da un Master e molteplici Chunkservers. Ogni file è diviso in chunk (pezzi). Ogni chunk ha un identificativo immutabile e globale di 64 bit chiamato chunk handle, assegnatogli al momento della creazione del file dal Master. I Chunkservers salvano in locale il chunk su linux file system, leggendo e scrivendo i chunks richiesti in base al chunk handle e un byte range. Per disponibilità e garanzie di persistenza ogni chunk è memorizzato su molteplici Chunkservers. Il Master contiene tutti i metadati del file system (namespace, access control list, mapping file/chunks e posizione in locale di ogni chunk) ed esegue le operazioni tipiche di un file system (gestione allocazione,

garbage collection e migrazione di chunks tra Chunkservers). Il Master comunica periodicamente con i Chunkservers per controllare il loro stato. Avere un singolo Master permette di avere una gestione ottimale della locazione dei chunks avendo il Master una visione globale del sistema. Onde evitare che il Master diventi un collo di bottiglia per le letture e scritture, il trasferimento dei chunks non è effettuato dal Master ma dai singoli Chunkservers che li contengono. Come mostrato in figura 2.6, il GFS Client utilizzando la dimensione fissa dei chunk, traduce il nome del file e byte offset specificato dal applicazione in un chunk index all'interno del file, poi manda al Master il nome del file e il chunk index. Il Master risponde con il chunk handle e l'indirizzo dei Chunkservers che contengono le repliche.

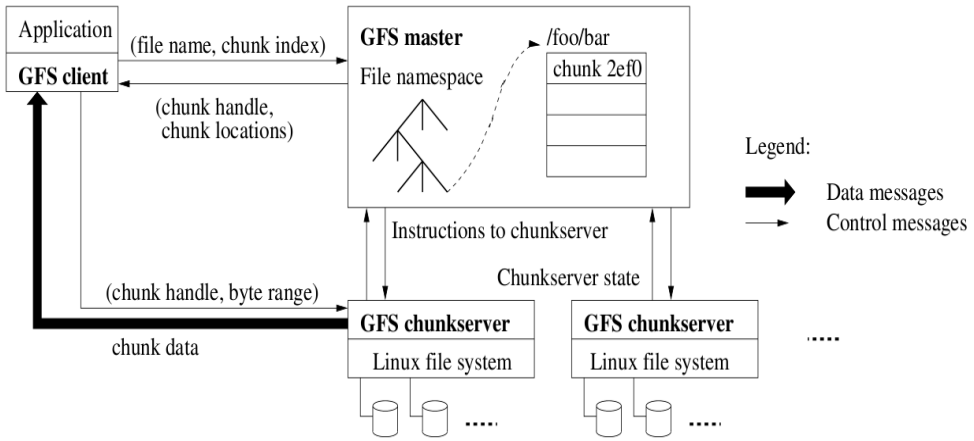


Figura 2.6: Google File System tratta da [8]

Dimensione dei Chunk

La dimensione dei Chunks è un parametro molto importante che fa variare di molto le performance del sistema. Il GFS utilizza una dimensione dei blocchi pari a 64 MB, molto maggiore rispetto a un tipico file system. Una tale dimensione offre molti vantaggi:

- la possibilità di fare cache del chunk index, permette al GFS Client di interagire una solo volta col Master agevolando le operazioni sul file di molti TB.
- riduzione dell'overhead aggiunto dalle connessioni TCP.
- riduzione metadati nel Master

Metadati

Il Master memorizza tre matadati per ogni file:

- file e chunk namespace
- mapping file/chunks
- locazione repliche chunk

I primi due sono salvati nel disco rigido del Master tramite operazioni di log che garantiscono persistenza ad eventuali crash del Master (journaling). La locazione dei chunk non è salvata sul disco rigido locale al Master, ma e chiesta ad ogni Chunkservers al momento dell avvio del Master o alla nuova unione di un Chunkservers ed tenuta in memoria principale per migliorare le performance.

Log

Le operazioni di log sono fondamentali in GFS, un operazione non è conclusa finchè non si è fatto il flush sul disco dei log su diverse macchine. Dopo un crash del Master, vengono eseguite tutte le azioni del log per riportare tutto allo stato precedente al crash. Quando i log crescono troppo si effettua un checkpoint, salvando i log su disco per minimizare le operazioni da effettuare dopo un crash.

Apache Hadoop HDFS[6] è la versione open source ispirata al Goolge File System.

2.9 Load Balancing

L'esecuzione di grossi calcoli in un architettura cluster, necessita di sistemi di suddivisioni del problema e monitoraggio dello stato dell'esecuzione per garantire correttezza dei risultati.

2.9.1 Google MapReduce

MapReduce si ispira alle funzioni map e reduce usate nella programmazione funzionale (LISP) e all'algoritmo Divide et Impera. La funzione di Map prende i dati in input del nodo principale e crea sotto problemi da inviare ai nodi operativi. I nodi operativo possono calcolare il problema mandatogli se è atomico, altrimenti suddividere il problema in ulteriori sotto problemi da spedire ad altri nodi operativi. Viene quindi a crearsi un albero multilivello. Le foglie dell'albero risolvono i problemi atomici e inviano i risultati

[6] hadoop.apache.org/hdfs/

ai genitori. La funzione Reduce prende i risultati dei problemi inviati e calcola una risposta totale.

Apache Hadoop MapReduce[7] è la versione open source ispirata al Google MapReduce.

2.10 Database management system distribuiti

Dal momento che le applicazioni sono sempre più data-intensive, neccesitano di database distribuiti, scalabili con performance ottimizzate per picchi di richieste.

2.10.1 Google BigTable

Come spiegato in [9], dalla necessità di archiviare svariati petabyte provenienti da Google Earth[8] e Goolge Finance[9] ed altre applicazioni, gli sviluppatori hanno creato la BigTable, un sistema di memorizzazione di strutture dati scalabile e distribuito. Il concetto alla base è quello di voler memorizzare delle pagine web, e di utilizzare l'URI[10] della pagine come chiave della riga e le informazioni relative alla pagine nelle colonne. In figura 2.7 mostra una riga con chiave l'uri al contrario con colonne con il contenuto della pagina e i link al suo interno. Ogni riga è rappresentata da una stringa al cui interno c'è una stringa con la chiave, stringhe per le colonne e un intero a 64 bit con il timestamp. Le righe sono ordinate in base alla chiave e dinamicamente raggruppate, cosi facendo le letture provenienti da una pagina web con lo stesso dominio saranno righe adiacenti.

[7] http://hadoop.apache.org/mapreduce/
[8] http://www.google.com/intl/it/earth/index.html
[9] http://www.google.com/finance
[10] http://it.wikipedia.org/wiki/Uniform Resource Identifier

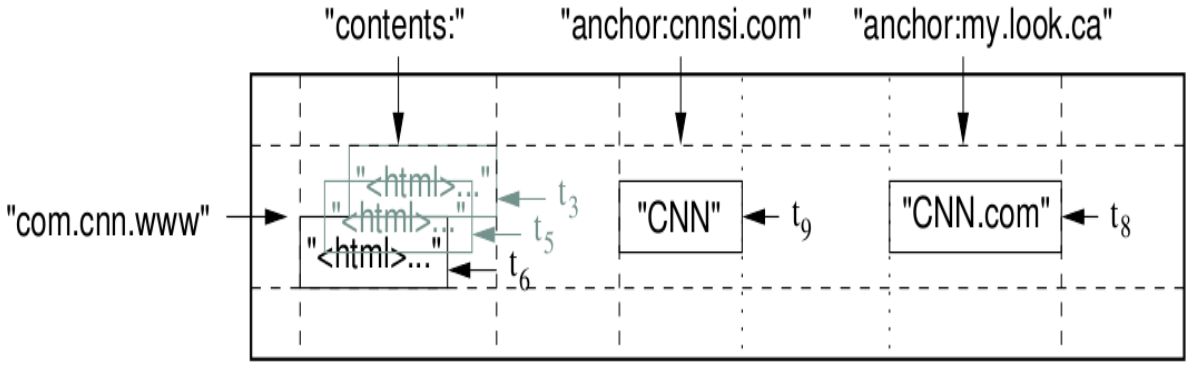


Figura 2.7: Esempi di una pagina salvata tratta da [9]

Apache Hadoop Hbase[11] è la versione open source ispirata alla BigTable.

2.11 Cloud Framework

Molti framework per cloud computing sono attualmente proprietari, non concedendo alla comunità scientifica di effettuare esperimenti. Nascono quindi framework open source che permettono di realizzare una IaaS su cluster, dando la possibilità di modificare il codice sorgente per effettuare test.

2.11.1 Eucalyptus

Come spiegato in [10] Eucalyptus[12] possiede un architettura semplice e modulare, con un design gerarchico che riflette la struttura di risorse in ambito accademico. Implementato con virtualizzazione XEN[13], ma in futuro si estenderà anche a KVM[14] e VMware[15]. Il team di sviluppo ha scelto di implementare ogni servizio di alto livello del sistema, come un web service a sè stante. I benefici di tale scelta sono:

- un servizio web espone API ben definite tramite un documento WSDL[16], che espone i dati i input/output per ogni operazione

[11] http://hbase.apache.org/
[12] http://www.eucalyptus.com/
[13] http://www.xen.org/
[14] http://www.linux-kvm.org/
[15] http://www.vmware.com/
[16] http://www.w3.org/TR/wsdl

- permette di utilizzare caratteristiche dei web service, come la comunicazione WS-security

L'installazione di EUCALYPTUS è composta da tre componenti in figura 2.8, ogni uno con la propria interfaccia.

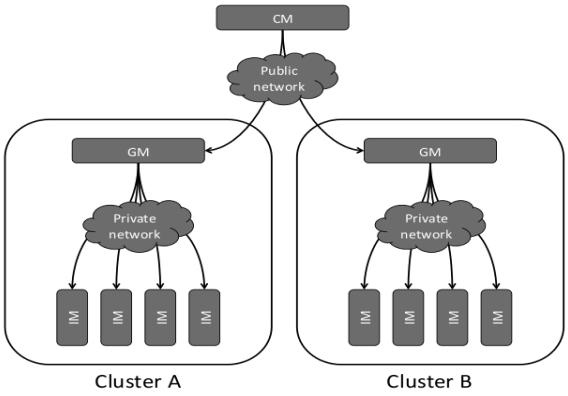


Figura 2.8: Struttura di Eucalyptus tratta da [10]

- **Instance Manager (IM):** è eseguito su ogni nodo designato ad ospitare macchine virtuali. L'IM interroga e controlla il sistema software del nodo (sistema operativo host e hypervisor) per rispondere alle richieste fattegli dal Group Manager (GM). L'IM scopre le risorse fisiche della macchina, (numero di core del processore, dimensione della memoria, e spazio libero su disco) per rispondere alle richieste describeResource e describeIstance fatte dal GM. Il GM decide l'esecuzione/terminazione di una VM, attraverso le richieste runInstance e terminateInstance fatte all'IM. Solo dopo la verifica della disponibilità effettiva delle risorse, e il controllo dei permessi (solo il padrone di una VM o l'amministratore puo terminare una VM), l'IM esegue la richiesta con l'aiuto dell'hypervisor.

- **Group Manager:** il GM è eseguito su una macchina che ha accesso al cluster che ospita gli IM. Le operazioni sono uguali a quelle di un IM ma su una pluralità di IM. Il GM ha tre funzioni principali:

  – eseguire lo scheduling attraverso le richieste runInstance effettuate sugli IM opportuni
  – controllare la rete tra le VM del gruppo
  – raccoglie informazioni sulle IM da mandare al cloud Manager per fornirgli una visione globale

  Non appena il GM riceve istruzioni sull'esecuzione di un insieme di VM, invia richieste di describeResource a tutte le macchine del gruppo, lanciando runInstance sulla prima macchina con risorse sufficienti. Per fornire una rete tra le macchine virtuali l'GM per sfuggire ai firewall utilizza il progetto Virtual Distributed Ethernet (VDE) [11] che simula il protocollo Ethernet tramite processi user-space che riproducono switch e cavi.

- **cloud Manager:** in figura 2.9 espone le risorse all'utente, è composto da tre livelli:

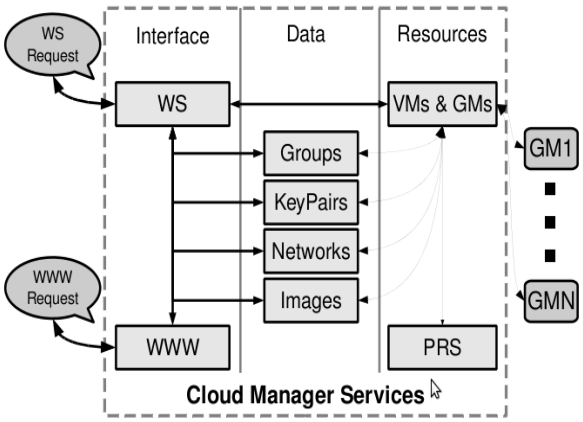


Figura 2.9: cloud Manager tratta da [10]

  - **Interface Services:** permette all'utente di autenticarsi e fornisce strumenti di gestione del sistema
  - **Data Services:** memorizza i dati di configurazione del sistema
  - **Resource Services:** permette la gestione delle operazione di allocazione e monitoraggio delli GM

2.12 Sfide alla ricerca

La tecnologia del cloud computing è stata ampiamente utilizzata dall'industria mondiale, ma la ricerca in questo campo è solo all'inizio. Ci sono questioni non completamente risolte e nuove problematiche emergono ogni giorno dall'industria delle applicazioni. In questa sezione illustreremo alcune sfide nel campo del cloud computing elencate in [6].

2.12.1 Allocazione automatica

Uno dei punti chiave del cloud computing è l'allocazione/deallocazione di risorse on-demand. L'obbiettivo dei cloud provider è di trovare sistemi di gestione delle macchine virtuali in modo da onorare contratti di Service Level Agreement (capitolo 3). Questo non è un problema nuovo, studi passati [12] [13] giungono alle seguente soluzione:

1. Costruzione di un modello di performance che predica quante risorse allocare in base al numero di richieste, in moda da garantire il rispetto della SLA

2. Periodiche previsioni sulle domande future

3. Allocazione automatica di risorse secondo i punti precedenti

Il modello di performance può essere costruito con varie tecniche come: Queuing theory [12], Control theory [14] and Statistical Machine Learning [15].

2.12.2 Migrazione macchine virtuali

La virtualizzazione fornisce significativi benefici attraverso la migrazione di macchine virtuali, che permette una distribuzione del carico su diversi data center. Recentemente la tecnica di migrazione e stata migliorata, dando la possibilità di migrazioni LIVE, facendo durare il processo di migrazione da pochi millisecondi ad un secondo. In [16] viene proposto di migrare un intero sistema operativo, permettendo di evitare le difficoltà della migrazione di processi.

2.12.3 Consolidamento server

La consolidazione dei server puo ridurre l'utilizzo di risorse e il consumo energetico. Tramite la migrazione di macchine virtuali si possono spostare le macchine virtuali poco utilizzate su una singola macchina fisica, ed attivare su essa politiche di risparmio energetico. Il problema dell'ottimizzazione del consolidamento consiste nel risolvere il problema del bin-packing[17] [17], che è un problema NP-completo, ma molte euristiche cercano di dare soluzione al problema [18] [19]. Il tempo utilizzato per il consolidamento non deve influire sulle performance del sistema. Sono necessari sistemi di risveglio delle macchine messe in risparmio energetico che non interferiscano con le performance delle applicazioni.

2.12.4 Gestione energetica

Il risparmio energetico è la questione più importante nell'ambito del cloud computing. Recenti studi [20] stimano il costo energetico e raffreddamento pari al 53% dei costi totali di un data center. Lo sviluppo di data center ad alta efficienza energetica ha suscitato forte interesse. Alcune soluzioni sono:

- riduzione della velocità delle CPU parziale spegnimento dei componenti [21].
- Scheduling che effettuano le proprie scelte consapevoli del consumo energetico e utilizzando la consolidazione dei server [22].

[17] consiste nel collocare n oggetti di dimensione variabile nel minor numero di contenitori possibile

- protocolli di rete energeticamente efficienti come la proposta del Energy Efficient Ethernet Task Force[18]

Lo scopo di questi studi e trovare un compromesso tra efficienza energetica e prestazioni.

2.12.5 Gestione e analisi del traffico di rete

L'analisi del traffico può permettere ottimizzazioni che migliorano l'esperienza dell'utente. Ci sono molte sfide aperte alla misurazione e analisi del traffico di rete, le soluzioni attuali analizzano le comunicazioni tra centinaia di nodi, ma in un data center modulare ci sono milioni di nodi. Gli strumenti attuali assumo che il traffico sia tra internet e il data center, tralasciando le connessioni interne al data center come MapReduce e File System Distribuiti. In [23] sono mostrate alcune tipologie di traffico, in modo da progettare correttamente l'infrastruttura di rete. In [24] viene eseguita un analisi del traffico interno al data center esaminando i percorsi dei router con Simple Network Management Protocol (SNMP).

2.12.6 Sicurezza dei dati

Dal momento che un SaaS provider non può accedere alle politiche di sicurezza del cloud provider, deve avere completa fiducia in esso. Il IaaS provider deve garantire confidenzialità (accessi controllati e sicurezza dei trasferimenti) e accertabilità (controllo dell'effettivo utilizzo dell'impostazione di sicurezza scelta). La prima è facilmente ottenibile con l'utilizzo della criptografia, mentre la seconda necessita di tecniche di attestazione remote. Queste tecniche consistono in un modulo per una piattaforma fidata dall'inglese Trusted Platform Module (TPM), che genera riepiloghi delle impostazioni non modificabile da terzi. In un ambiente virtualizzato come quello del cloud computing, in cui con la migrazione live si trasferiscono macchine virtuali da una sorgente ad una destinazione, i sistemi di attestazione remota non sono sufficienti. Bisogna implementare meccanismi fidati ad ogni livello dell'architettura. Come proposto in [25] bisogna usare hardware TPM ed un monitor di virtualizzazione fidato. Inoltre la migrazione di macchine virtuali deve avvenire tra sorgenti e destinazioni fidate.

2.12.7 Piattaforme software

Il cloud computing fornisce piattaforme per applicazione scalabili data-intensive. Recenti studi hanno dimostrato che le performance e il consumo di risorse di un MapReduce Job dipendono fortemente dal tipo di applicazione [26]. Ad esempio un sort è fortemente legato all'I/O mentre un grep necessetà di significativa potenza di CPU. Inoltre dal momento che i nodi MapReduce hanno caratteristiche eterogenee e dipendenti dal numero

[18] http://www.ieee802.org/3/az/

di VM su una macchina fisica, è possibile migliorare le prestazioni facendo attenzione alle configurazione delle macchine virtuali [27] ed usando algoritmi di scheduling ottimizzati [28].

2.12.8 Tecnologie di archiviazione dati

Software Framework come MapReduce e sue varianti (Hadoop MapReduce e Dryad[19]) sono progettate per suddividere elaborazioni su grandi insiemi di dati. Come menzionato precedentemente i file systems distribuiti, sono differenti da quelli tradizionali nella loro struttura di archiviazione, modalità di accesso e API. In particolare non implementano le interfacce POSIX[20] per lo scambio di file tra diversi sistemi operativi, quindi introducono problemi di compatibilità. Diversi studi sono stati fatti su questo problema. In [29] viene proposto un metodo di supporto a MapReduce, usando file system per clust come IBM GPFS, mentre in [30] vengono proposte API primitive per un accesso scalabile e concorrente ai dati.

2.12.9 Nuove architetture di cloud

Come precedentemente detto, i grandi data center creano economie di scala, ma richiedono grandi quantità di energia e forti investimenti iniziali. Recenti studi [31] [32] suggeriscono che piccoli data center sono molto più vantaggiosi sotto certi aspetti:

- non consumano grandi quantità di energia e non richiedono grandi sistemi di raffreddamneto
- sono più economici da costruire favorendo una migliore geo-localizzazione con vantaggi di latenza

In [33] viene usato hardware fornito da volontari per ospitare applicazioni cloud.

[19] http://research.microsoft.com/en-us/projects/Dryad/
[20] http://standards.ieee.org/develop/wg/POSIX.html

Capitolo 3

Service Level Agreement

3.1 Cos'è un contratto di SLA

L'interesse verso le applicazioni web è in forte crescita. Nasce quindi la necessità di garanzie di Qualità di Servizio (QoS). Basta pensare ad applicazioni real-time come VoIP e Videoconferenza. Dove la mancanza di prestazioni adeguate, rende vano l'utilizzo del servizio stesso. Nel caso del VoIP i parametri di QoS vengono rappresentati dal numero massimo di pacchetti persi, che non deve superare il 5%. Con valori superiori la comunicazione tra le parti risulterebbe incompleta, facendo provare frustrazione agli utenti, dovuta all'impossibilità di parlare.

3.2 SLA Framework

Come spiegato in [34], gli obbiettivi di uno SLA framework sono due:

- fornire processi che gestiscano in modo automatico le interazioni business-to-business (B2B), per favorire l'incontro tra domanda e offerta.
- fornire processi che controllino l'adempimento degli obiettivi descritti nel contratto.

3.3 WSLA Framework

WSLA[1] è un'architettura proposta e sviluppata da IBM[2] per la specifica e il monitoraggio di Service Level Agreement per Web Services. Di seguito elencheremo le metriche di valutazione che compongono i servizi del framework.

[1] http://www.research.ibm.com/wsla/
[2] http://www.ibm.com/it/it/

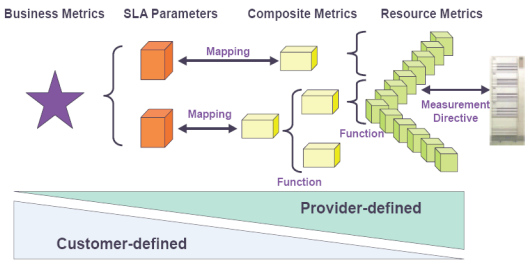


Figura 3.1: Livelli di gestione delle informazioni tratta da [34]

- **Metriche delle risorse (resource metrics):** ricavate direttamente dalle risorse gestite dal cloud provider (routers, server, strumentazioni varie). Integrate in WSLA tramite direttive di misurazione, assegnate ad ogni risorsa per tenere traccia del contesto e dei comandi necessari per eseguire la misura.
- **Metriche composite (composite metrics):** composizione di metriche base (o composte a loro volta) tramite opportuni algoritmi. L'operazione di composizione è solitamente svolta dal cloud provider, ma può essere affidata anche ad una terza parte.
- **Parametri SLA (SLA parameters):** parte fondante di un contratto di SLA, rappresenta l'interfaccia tra il cloud provider e il SaaS provider di un servizio. Deve definire tutti i parametri, insieme con i rispettivi intervalli di validità e con i mapping verso una metrica opportuna. Deve essere possibile specificare se la valutazione di un parametro debba avvenire quando questo soddisfi o meno un obiettivo a livello di servizio. La valutazione dei parametri può essere delegata ad una terza parte, per garantirne l'obiettività.
- **Metriche aziendali (business metrics):** sono la base della strategia di gestione del rischio dei clienti e restano interne ad essi. Esistono metriche aziendali anche per il fornitore del servizio, che deve verificare che i parametri di SLA dei servizi forniti non vadano in conflitto con le proprie politiche aziendali.

3.3.1 Obiettivi del progetto

Grazie al design molto flessibile il framework WSLA può definire parametri di SLA in svariati contesti, costruendo le metriche composite in base al contesto applicativo. La sua modularizzazione permette di definire terze parti per il monitoraggio delle prestazioni nel rispetto degli accordi tra le due parte del contratto. La figura 3.2 mostra fornitori multipli di servizi, con al centro servizi di auditing e monitoraggio effettuati da terze parti. Ogni fornitore di servizi conosce solamente la parte di contratto che gli compete, secondo il principio di conoscenza minima (need to know principle). Questo principio

garantisce la privacy delle parti e permette una più semplice sostituzione dei fornitori di servizi nel caso ce ne sia bisogno.

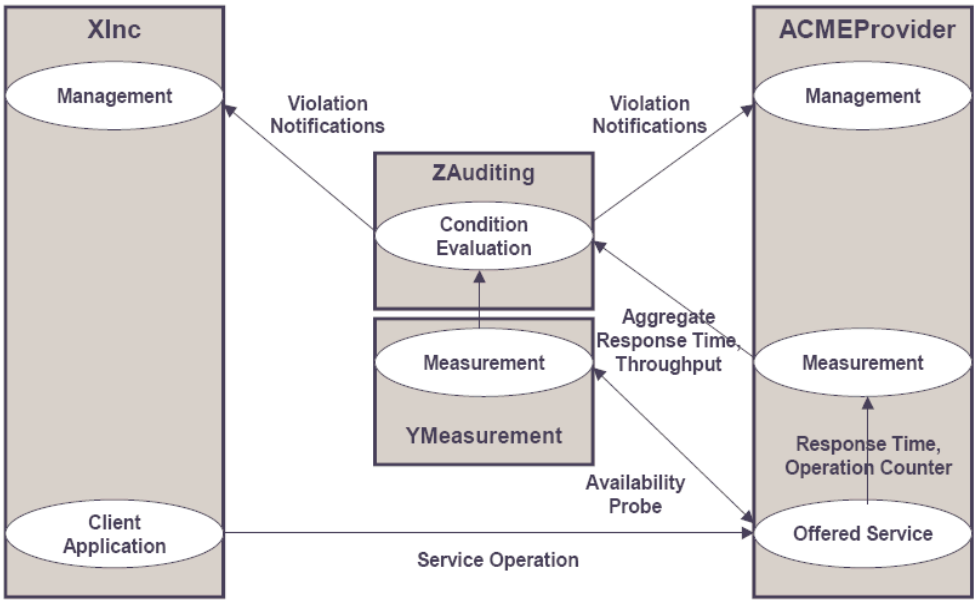


Figura 3.2: Interazioni tra le parti tratta da [34]

3.3.2 Le fasi di WSLA

Premesso che WSLA può essere utilizzato anche al di fuori del contesto delle applicazioni web, supponiamo che il contratto di SLA sia definito per un servizio web definito da un documento di Web Services Description Language (WSDL)[3] pubblicato su un indice pubblico come l'Universal Description Discovery e Integration (UDDI)[4]. La figura 3.3 mostra le fasi dalla stipulazione di un contratto alla sua terminazione .

[3]http://www.w3.org/TR/wsdl
[4]http://www.uddi.org

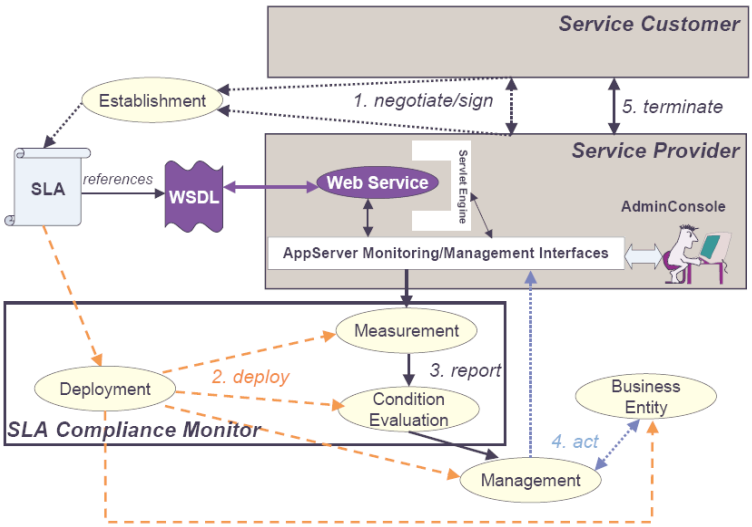


Figura 3.3: Fasi di un contratto di SLA tratta da [34]

1. **Negoziazione del SLA:** consiste nell'incontro della domanda e l'offerta. Le parti in causa si accordano sulla qualità del servizio e il prezzo. Il cloud provider valuta le esigenze del cliente e può imporre vincoli sulle richieste in arrivo (es. massimo 100 richieste/secondo). Questa fase si conclude con la firma da parte delle parti di un contratto di SLA.

2. **Sviluppo del SLA:** questa fase consiste nella suddivisione dei ruoli, da consegnare alle parti esterne incluse nel contratto. Sempre per il principio di minima conoscenza, si creano interfacce per la cooperazione tra le parti limitando la visione completa del contratto.

3. **Misurazione e valutazione dei livelli di servizio:** il servizio di misurazione raccoglie le misurazioni provenienti dalle risorse, per la verificare il rispetto degli accordi definiti nel contratto. In particolare, il servizio di misurazione tiene traccia della configurazione del sistema e delle informazioni di run-time sulle metriche coinvolte. Mentre il servizio di Valutazione confronta i dati raccolti con i limiti imposti nel contratto, avvertendo le parti in causa qualora vi siano violazioni.

4. **Azioni correttive:** una volta rilevata una violazione di un qualche obiettivo, viene notificata con un messaggio alla Business Entity, che è un entità aziendale ad esempio un dipendente o un sitema di Enterprise Resource Planning (ERP) che applicherà le eventuali azioni correttive.

5. **Terminazione del SLA:** è possibile definire condizione che fanno decadere il contratto tra le parti. Ad esempio la violazione di parametri importanti per il SaaS provider o la cessazione naturale per scadenza del contratto.

3.3.3 Il linguaggio WSLA

Il linguaggio WSLA è definito da uno schema XML Schema, in questo paragrafo esamineremo le componenti principali.

- **Soggetti:** la prima sezione di un documento WSLA definisce tutte le parti coinvolte nell'accordo, i firmatari (SaaS provider e cloud provider) e le terze parte che si occupano dei servizi accessori.

- **Servizi:** la seconda sezione del documento descrive il servizio in termini di parametri osservabili. Il componente di misurazione accede a questa parte per effettuare le misurazioni. Le informazioni qui contenute servono a descrivere i parametri della SLA, ovvero definire delle metriche per le performance dei vari servizi. Gli oggetti per i quali si definiscono le metriche sono le unità atomiche su cui il servizio lavora. Nel caso di applicazioni web, corrispondono alle singole operazioni espresse nel WSDL. La figura 3.4 rappresenta un oggetto (ServiceObject) con un operazione (getQuote) con parametri espressi con metriche composte.

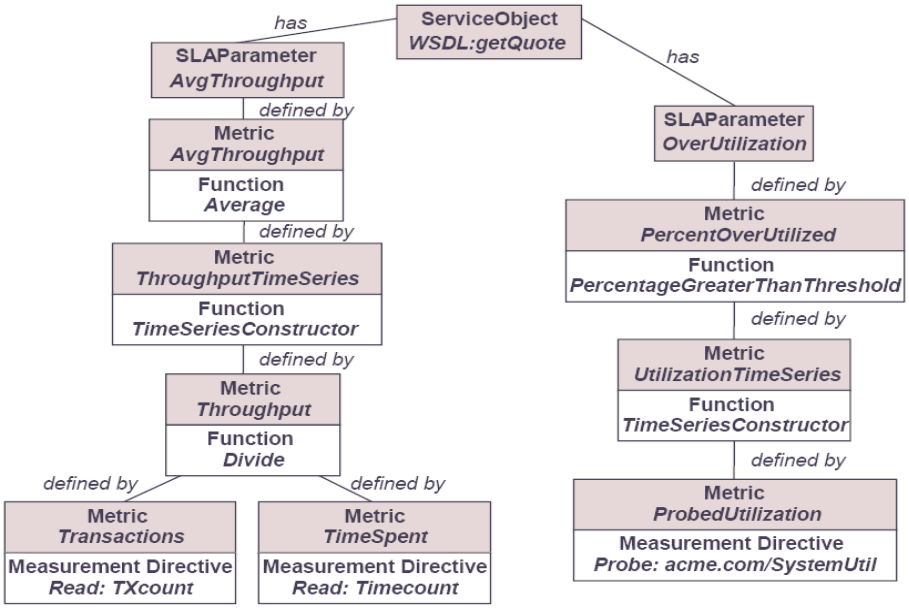


Figura 3.4: Esempio di descrizione di un servizio tratta da [34]

- **Obblighi:** La terza sezione del documento WSLA descrive i vincoli e le garanzie che devono essere rispettati dai parametri del servizio. Il componente di valutazione delle condizioni accede a questa parte per effettuare le verifiche. Esistono due diversi tipi di obbligazione che le parti assumo di rispettare:

- **Service Level Objective (SLO):** Gli obiettivi a livello di servizio rappresentano una sorta di "promessa" circa lo stato di un parametro. Solitamente è il cloud provider a farsi carico del mantenimento di un SLO, anche se esistono casi in cui è il SaaS provider a dover garantire il mantenimento di un certo obiettivo.
- **Action Guarantees:** le garanzie di azione consistono nell'esecuzione di una determinata operazione al verificarsi di una condizione.

3.3.4 Implementazioni di WSLA Framework

Il primo prototipo del framework WSLA è stato introdotto nel Web Services ToolKit, un ambiente integrato di progettazione, sviluppo e gestione di web services realizzato da IBM. Recentemente, il toolkit si è evoluto in una nuova versione, ed è diventato parte di un sistema più ampio denominato Emerging Technologies ToolKit[5], di cui costituisce il componente ETTK-WS. Il progetto ETTK vuole riunire in un unico ambiente tutte le tecnologie emergenti in fase di sviluppo, e il tema dei SLA rientra in questa categoria. La versione più recente di ETTK-WS vede l'abbandono del formato WSLA a favore di WS-Agreement, uno standard che rappresenta l'evoluzione di WSLA e che oltre all'IBM vede coinvolti importanti soggetti come il W3C[6], Hewlett-Packard[7] e altri. La volontà da parte delle grandi società, a cooperare verso il raggiungimento di un accordo comune in temi nuovi e importanti come quello dei SLA, fa ben sperare per il futuro di queste tecnologie.

3.4 WS-Agreement Framework

WS-Agreement è la proposta del Global Grid Forum per la creazione e il monitoraggio d'accordi fra cloud provider e SaaS provider. Permette ai primi di valutare le proprie risorse e in base alle proprie potenzialità di accettare o rifiutare la richiesta per un accordo. I secondi possono esprimere i loro requisiti sulla QoS in un formato formale (XML[8]), comprensibile dalla macchina.

[5] http://www.alphaworks.ibm.com/ettk

[6] http://www.w3.org/

[7] http://www8.hp.com/

[8] eXtensible Markup Language è un metalinguaggio di markup, ovvero un linguaggio marcatore che definisce un meccanismo sintattico che consente di estendere o controllare il significato di altri linguaggi marcatori

3.4.1 Struttura di Accordi e Template

Gli accordi fra cloud provider e SaaS provider si specificano in un linguaggio basato su XML e si conformano alla struttura presentata in figura 3.5. Un accordo ha una struttura simile a quella del template. La specifica WS-Agreement fornisce solo la struttura di accordi e template, mentre l'implementazione dipende dal dominio del fornitore.

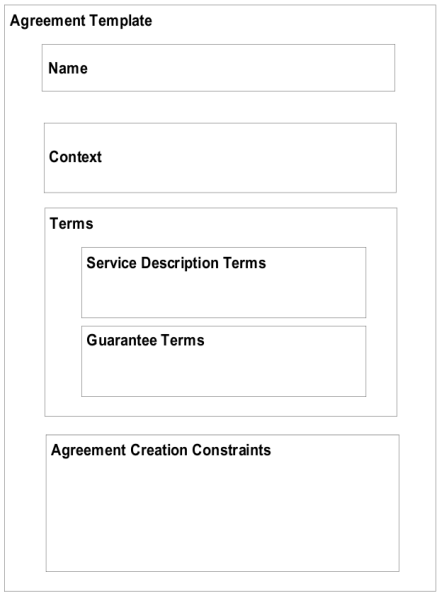


Figura 3.5: Struttura di WS-Agreement Template tratta da [34]

- **Context:** contiene informazioni (metadati) sull'accordo stesso: i partecipanti coinvolti, il template di riferimento per creare un nuovo accordo, il tempo di validità dell'accordo e riferimenti ad altri accordi associati.
- **Service Description Terms (SDT):** specifica le operazioni del servizio che saranno eseguite se la conformità all'accordo sarà verificata. La definizione delle operazioni all'interno di questa sezione e la loro corrispondenza ad un servizio reso disponibile dal cloud provider può essere espressa tramite riferimenti formali, in qualsiasi linguaggio accettato o non formali. La specifica non definisce il linguaggio per descrivere i riferimenti finali al servizio, essa fornisce solo la rappresentazione astratta che può essere implementata con elementi standard o specifici del dominio.
- **Guarantee Terms (GT):** definisce la modalità di fornitura delle operazioni del servizio. In questa sezione sono riportati i Service Level Objectives (SLO) (ad esempio il tempo medio di risposta per i servizi descritti nelle sezioni SDTs), le condizioni che devono essere rispettate perché un dato SLO possa essere fornito (ad

esempio, il tasso di richiesta deve essere inferiore ad una soglia predefinita) e un valore aziendale che rappresenta l'importanza di raggiungere l'obiettivo descritto dal SLO nell'accordo (ad esempio, un valore aziendale alto implica che il fornitore usi più risorse per eseguire le operazioni descritte nella sezione SDT, nel caso in cui le risorse disponibili non possano soddisfare il SLO definito).

- **Constraint:** si presenta solo nella struttura di un template e ha lo scopo di imporre vincoli ai valori accettati per riempire i campi dei termini presentati sopra. Per esempio, il valore accettato per il numero di CPUs usate per soddisfare un SLO, non può avere valore più di 7. Offerte di accordi che richiedono un numero di CPUs più di 7 non saranno accettate. Questa sezione si usa per verificare la conformità di una richiesta per la creazione di un accordo con un dato template.

3.4.2 Modello di Interazione fra Accordi

WS-Agreement fornisce un protocollo di trattative necessarie per creare nuovi accordi. Le parti coinvolte sono due. I partecipanti hanno il ruolo dell'Agreement Initiator (AI) o del'Agreement provider (AP), indipendentemente dai ruoli che hanno nelle interazioni fra servizi. Quindi l'AI può essere sia il cloud provider sia il SaaS provider. Il modello di interazione figura 3.6 specifica solo la creazione di accordi in base a un template. L'AI espone le operazioni createAgreement(A) e getTemplates(). Il componente denomi-

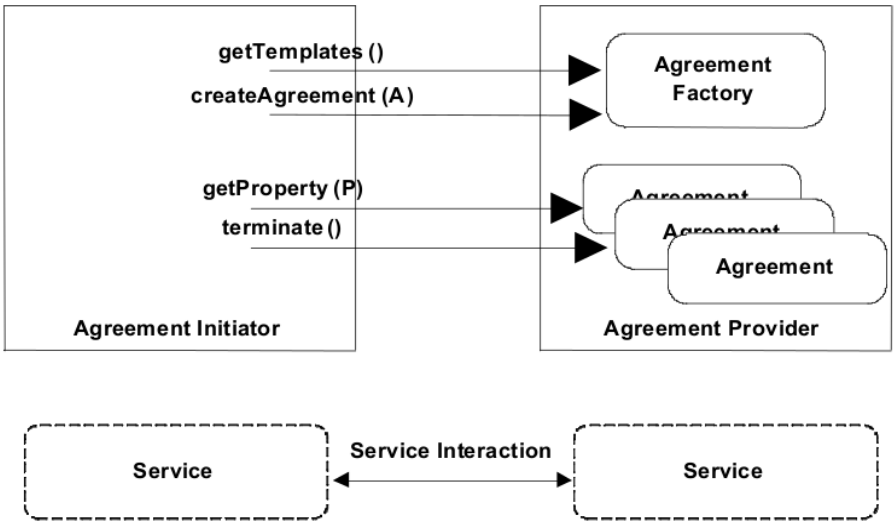


Figura 3.6: Modello interazioni WS-Agreement tratta da [34]

nato Agreement Factory risponde all'invocazione di queste operazioni, rispettivamente, creando nuovi accordi e recuperando i template disponibili. L'ordine dell'invocazione di queste operazioni è la seguente: l'AI recupera i template proposti dal AP chiamando

l'operazione getTemplates(), e crea un'offerta riempiendo i campi del template scelto dalla lista precedente tramite createAgreement(A). Se l'offerta viene accettata dal AP, un nuovo accordo si crea. Il modello definisce i "port type" per il monitoraggio del servizio e dello stato delle garanzie del nuovo accordo. Il monitoraggio di questi termini viene fatto mediante l'operazione getProperty(P). In particolare, i servizi descritti nella sezione SDTs dell'accordo possono essere in stato NotReady (l'esecuzione del servizio non è stata iniziata), Ready (il servizio è pronto per essere eseguito), Processing (il servizio è in esecuzione) e Completed (l'interazione con il servizio è stata completata). Invece, le garanzie fornite dall'accordo possono essere in stato Fulfilled, Violated e NotDetermined. Questi valori indicano se una garanzia è stata mantenuta, disattesa o la sua verifica è in corso. In questo modo, si verifica l'adempimento del contratto durante l'esecuzione del servizio.
WS-Agreement specifica il modello astratto di interazione fra l'iniziatore e il fornitore, con le seguenti limitazioni:

- Non è un protocollo completo di trattative. Deve essere esteso per fornire operazioni più complesse. Ad esempio una procedura di trattativa delle condizioni dei termini per creare un accordo.
- Non definisce se il monitoraggio viene fatto da terze parti o dal SaaS provider.

3.4.3 Cremona: Architettura di WS-Agreement

Nello scenario di interazioni fra servizi Web basate su accordi, tanti argomenti rimangono in sospeso. La specifica WS-Agreements definisce la struttura e il modello necessario per descrivere e scambiare i requisiti del SaaS provider e la potenzialità del cloud provider. Tuttavia, non fornisce l'implementazione dei termini dell'accordo (SDT e GT). L'architettura Cremona, sviluppata da IBM, definisce questi componenti ed implementa le funzionalità per il supporto del protocollo WS-Agreement e per il monitoraggio delle interazioni basate su accordi.

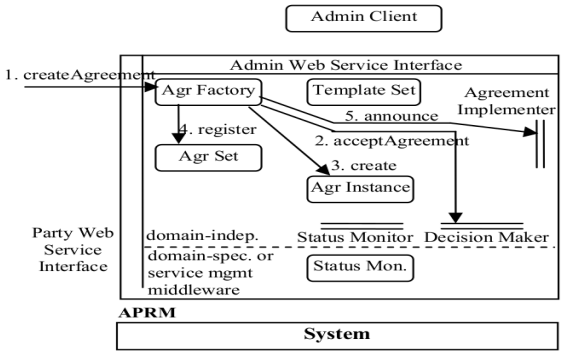


Figura 3.7: APRM struttura del fornitore dell'accordo tratta da [34]

L'architettura si compone di tre livelli. Al primo livello in figura 3.7, un insieme di funzioni chiamato Agreement Protocol Role Management (APRM) implementa il protocollo WS-Agreement. Di conseguenza, tutte le operazioni e i "port type"del protocollo vengono definiti per entrambi i partecipanti dell'accordo. L'obiettivo dell'APRM è lo scambio dei requisiti del SaaS provider e delle potenzialità del cloud provider finchè tutti e due sono d'accordo sui termini del contratto. Per raggiungere questo obiettivo, dalla parte del fornitore sono presenti i componenti necessari per la creazione di contratti in base a un template: Agreement Factory, Template Set, Agreement Set e Agreement Instance. Inoltre, l'accettazione o il rifiuto di un'offerta per un accordo richiede il monitoraggio dei servizi corrispondenti. Il fornitore deve essere in grado di analizzare le potenzialità delle risorse del sistema e valutare l'esecuzione dei servizi richiesti. Questo si realizza tramite il componente Status Monitor Implementation. Dalla parte dell'iniziatore dell'accordo, l'obiettivo del componente APRM è il recupero dei template dal fornitore, la creazione di un'offerta per un accordo, la richiesta per un'offerta al fornitore, e l'archivio degli accordi accettati. La realizzazione di queste operazioni si raggiunge con i componenti Agreement Initiator, Factory Set, Template Processor, e Agreement Set.

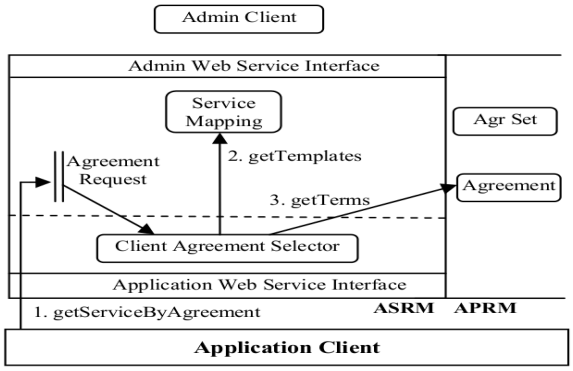


Figura 3.8: ASRM struttura dell'iniziatore dell'accordo tratta da [34]

Il secondo livello, figura 3.8, si chiama Agreement Service Role Management (ASRM) ed è composto da funzionalità per collegare i termini di un accordo al sistema di servizi. Questo livello fornisce tutti i meccanismi necessari al fornitore per creare template e valutare offerte di accordi in base alle capacità di servizi. Permette il monitoraggio delle proprietà di servizi durante la loro esecuzione e azioni da intraprendere per assicurare i SLO in un accordo. Dalla parte dell'iniziatore, l'ASRM implementa i componenti necessari per il recupero di servizi che soddisfano i suoi requisiti, per ricavare i template dei servizi, e per creare un'offerta per un accordo da un template scelto dalla lista.

L'ultimo insieme di funzioni si chiama Strategic Agreement Management (SAM), esso fornisce le informazioni per implementare i livelli APRM e ASRM. I suoi componenti osservano e misurano l'ambiente dei servizi, e dai risultati ricavati stabiliscono quali template debbano essere modificati e quale sia l'entità delle loro modifiche. L'architettura di Cremona attualmente non fornisce questo set di funzionalità.

Nell'ultima versione di ETTK dell'IBM il sistema WSLA e sostituito da WS-Agreement.

3.5 Confronto tra WSLA e WS-Agreement

I sistemi WSLA e WS-Agreement risolvono in modo molto simile le problematiche legate ai Service Level Agreement. Lo schema dei documenti che descrivono gli accordi è molto simile. Entrambi prevedono uno spazio per la descrizione dei soggetti coinvolti e dei servizi forniti, insieme agli indici di qualità per ogni servizio. Così come WSLA, anche WS-Agreement permette di coinvolgere terze parti nella fornitura dei servizi e soprattutto nelle fasi di monitoraggio dei parametri dell'accordo. A differenza di WSLA, in WS-Agreement le terze parti sono nascoste da un'unica interfaccia. WSLA è orientato alla definizione di garanzie che devono essere fornite per l'adempimento di un accordo.

Il linguaggio specifica queste garanzie sugli attributi di servizi. Dall'altra parte, WS-Agreement definisce accordi ad un livello diverso dal livello del servizio ed è orientato alla gestione di accordi (creazione, monitoraggio). Infatti, gli accordi in WS-Agreement vengono definiti indipendentemente dalla struttura dei servizi. Il vantaggio di questo approccio sta nel fatto che in WS-Agreement è possibile definire accordi su servizi esistenti (servizi generici e non solo di natura Web) senza cambiare la loro implementazione (indipendenza alle modifiche). Ciò che differenzia maggiormente WS-Agreement da WSLA è la definizione di template e di Agreement Factory; queste due entità permettono di definire meglio le interazioni tra le parti per la stipula di un contratto, operazione che non era formalmente specificata nella descrizione del framework WSLA. Inoltre, WS-Agreement definisce solo le interfacce degli accordi, lasciando totale libertà ai fornitori dei servizi per quanto concerne la loro implementazione. WSLA invece definiva più formalmente la struttura delle metriche per la misurazione dei parametri. Una caratteristica di WSLA non riportata in WS-Agreement è la distinzione tra Service Level Objectives e Action Guarantees, che permetteva una più accurata definizione degli accordi. I SLO di WS-Agreement, inoltre, possono contenere un indice di priorità che dà una stima del valore aziendale di un dato obiettivo. Questa scelta è in un certo senso contraria al principio di isolamento delle metriche di Business perseguito dal sistema WSLA, che isolava nel modulo chiamato "Business Entity" tutte quelle variabili della strategia aziendale. Naturalmente, entrambi gli approcci (rendere noto il valore relativo di un obiettivo o nasconderlo completamente) presentano vantaggi e svantaggi. Banalmente, conoscere le diverse priorità degli obiettivi che compongono un accordo permette di reagire con più prontezza alle violazioni degli obiettivi più critici, mentre nascondere tutte le politiche aziendali dietro un modulo opaco può essere la scelta migliore nel caso di aziende con una identità molto forte da preservare. Del resto, l'indice prioritario previsto per gli SLO in WS-Agreement è solo opzionale, e ciò permette di venire incontro ad entrambe le esigenze.

Capitolo 4

Soluzioni esistenti

In questo capitolo analizzeremo le tre migliori soluzioni di hosting su architettura cloud.

4.1 Google AppEngine

Google AppEngine[1] (GAE) permette di eseguire le proprie applicazioni nella potente e distribuita rete di cloud computing di google. GAE rende semplice lo sviluppo e il mantenimento di applicazione fornendo svariati servizi:

- archiviazione persistente con query, ordinamento e transazioni
- auto scaling e load balancing automatico
- API per autenticazione e utilizzo delle google app come gmail, google maps ecc.
- SDK per sviluppo in locale
- coda di operazioni per effettuare operazioni di gestione che non riguardano i servizi dell'applicazione

GAE offre due ambienti di sviluppo **Java** e **Python**.

L'ambiente **Java** utilizza le tecnologie tipiche del mondo Java come: Java Servlet e Java Server Page o altri linguaggi JVM-compatibili come JavaScript e Ruby. La versione di java e la 6, ma supporta anche applicazioni scritte con la versione 5 con le dovute operazioni di refactoring. Dal momento che le applicazioni per motivi di sicurezza girano in sandbox[2], (che fornisce un ambiente limitato, senza accesso al sistema

[1] http://code.google.com/intl/it-IT/appengine/

[2] ambiente di test, spesso slegato dal normale flusso di ambienti predisposti per lo sviluppo ed il test delle applicazioni

operativo sottostante) per cui alcune operazioni come aprire un socket o scrivere sul file system comportano eccezioni, bloccando l'esecuzione dell'applicazione. Le API di GAE forniscono molti servizi come:

- Java Data Objects (JDO) and Java Persistence API (JPA) per il trattamento dei dati salvati
- JavaMail per invio di email
- Java Net HTTP per il fetch di risorse al di fuori del dominio di google utilizzando il servizio di URL fetch
- API per le app di google come Google Maps, Docs, Youtube ecc.

Tra i tools di sviluppo troviamo:

- **Google Web Toolkit:** permette di scrivere l'interfaccia grafica in Java e convertirla in JavaScript per l'utilizzo in Browser. Il grado di compatibilità multi browser dell'interfaccia è molto elevato, garantendo lo stesso comportamento.
- **Speed Tracer:** estenzione per il browser Google Chrome, è possibile fare analisi di prestazione per applicazioni rich client, potendo cosi valutare soluzioni algoritmiche differenti per migliorare l'esperienza utente.
- **GWT Design** è possiblie disegnare in maniera grafica l'interfaccia e automaticamente convertirla in Java. Questo tool permette di progettare interfacce grafiche molto sofisticate con caratteristiche come Drag and Drop e internazionalizzazione (i18n).

Questi tools sono integrati in un plugin del ambiente di sviluppo Eclipse[3].

L'ambiente **Python** permette di sviluppare applicazioni in python con tecnologia Common Gateway Interface (CGI)[4]. Non offre molti tools agli sviluppatori, in quanto è possibile utilizzare qualsiasi web framework come Django[5], web2py[6] e altri. GAE offre un piccolo framework chiamato webapp[7], che semplifica la gestione delle richieste e la compilazione delle risposte (template). Per l'intefaccia grafica si usano i normali framework JavaScript come Ext JS[8], JQuery[9] o grazie a Pyjamas[10] è possibile scrivere l'interfaccia

[3] http://www.eclipse.org/
[4] http://it.wikipedia.org/wiki/Common_Gateway_Interface
[5] http://www.djangoproject.com/
[6] http://www.web2py.com/
[7] http://code.google.com/intl/it-IT/appengine/docs/python/tools/webapp/
[8] http://www.sencha.com/
[9] http://jquery.com/
[10] http://pyjs.org/

in python e convertirla in Javascript compatibile e ottimizzato. L'SDK Python permette lo sviluppo locale simulando il datastore e l'upload nei data center GAE.

DataStore

Uno dei servizi comuni ai due ambienti (sebbene con API diverse) è il DataStore. Implementato sui precedentemente citati GFS e la BigTable fornisce uno strumento di archiviazione robusto, altamente scalabile e con forti ottimizzazioni sulle performance delle query. Il DataStore fornisce due modalità:

- **Master/Slave:** consiste in un sistema di master e slave distribuito tra datacenter. Le operazioni di scrittura vengono effettuate tutte sul master, che asincronicamente replicherà le operazioni sugli slave in altri data center. Questa soluzione garantisce un elevata disponibilità in lettura, ma temporanee negazioni di scrittura dovuti a periodici downtime (manutenzioni programmate) del data center in cui e presente il master.
- **High Replication:** consiste nella replicazione delle operazioni su diversi data center secondo l'algoritmo di Paxos [35]. Questa soluzione non soffre dei downtime, ma di un'elevata latenza dovuta alla replicazione.

La figura 4.1 mostra le differenze di performance e consumo di risorse delle due soluzioni.

	Master/Slave	High Replication
Cost		
Storage	1x	3x
Put/Delete CPU	1x	3x
Get CPU	1x	1x
Query CPU	1x	1x
Performance		
Put/Delete Latency	1x	1x–2x
Get Latency	1x	1x
Query Latency	1x	1x
Consistency		
Get/Put/Delete	Strong	Strong
Most Queries	Strong	Eventual
Ancestor Queries	Strong	Strong
Occasional Planned Read-Only Period	Yes	No
Unplanned Downtime	Rare. Possible to lose a small % of writes that occurred near the downtime (recoverable after event).	Extremely rare. No data loss.

Figura 4.1: Differenza di performance tratta dal sito di AppEngine

Le interrogazioni avvengono con il linguaggio GQL (Google Query Language) in figura 4.2, con sintassi basata su SQL[11], ma non ancora sviluppato completamente. La sintassi in figura 4.2:

```
SELECT [* | __key__] FROM <kind>
  [WHERE <condition> [AND <condition> ...]]
  [ORDER BY <property> [ASC | DESC] [, <property> [ASC | DESC] ...]]
  [LIMIT [<offset>,]<count>]
  [OFFSET <offset>]

<condition> := <property> {< | <= | > | >= | = | != } <value>
<condition> := <property> IN <list>
<condition> := ANCESTOR IS <entity or key>
```

Figura 4.2: Sintassi di GQL tratta dal sito di AppEngine

Ogni query inizia con Select *, non permettendo la selezione parziale delle proprietà. Sono numerose le funzionalità tipiche di un linguaggio interrogativo ancora in sviluppo.

BlobStore

La dimensione massima degli oggetti inseribili nel DataStore è di 2 GB. Per file di dimensioni maggiori occorre utilizzare il servizio di BlobStore, che permette tramite un form html di inviare file come video e immagini, che possono essere serviti dall'applicazione tramite la blob-key, un identificatore univoco del blob[12] (Binary large object).

[11] http://it.wikipedia.org/wiki/SQL
[12] http://it.wikipedia.org/wiki/Binary_large_object

Quota Free e Prezzi

GAE fornisce una quota di risorse giornaliere gratuite, superata questa soglia le risorse costano i prezzi in figura 4.3. L'amministratore dell'applicazione può assegna un budget giornaliero, e suddividere questo budget in base alla risorse che più utilizza.

Resource	Unit	Unit cost
Outgoing Bandwidth	gigabytes	$0.12
Incoming Bandwidth	gigabytes	$0.10
CPU Time	CPU hours	$0.10
Stored Data	gigabytes per month	$0.15
High Replication Storage	gigabytes per month	$0.45
Recipients Emailed	recipients	$0.0001
Always On	N/A (daily)	$0.30

Figura 4.3: tratta dal sito di AppEngine

4.2 Microsoft Azure

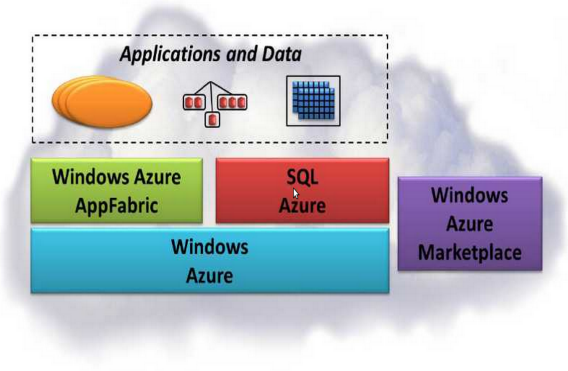


Figura 4.4: Struttura Azure tratta dal sito di Microsoft Azure

Anche Microsoft è scesa nel compo del cloud computing, fornendo Microsoft Azure[13] un servizio di PaaS. Il servizio di Microsoft è formato da 4 componenti in figura 4.4:

[13]http://www.microsoft.com/windowsazure/

- **Microsoft Azure:** fornisce un ambiente windows per gestire set di risorse provenienti da un data center. Astrae l'utente dalla complessità dell'infrastrutture, dandogli la possibilità di storage e potenza di calcolo on-demand.

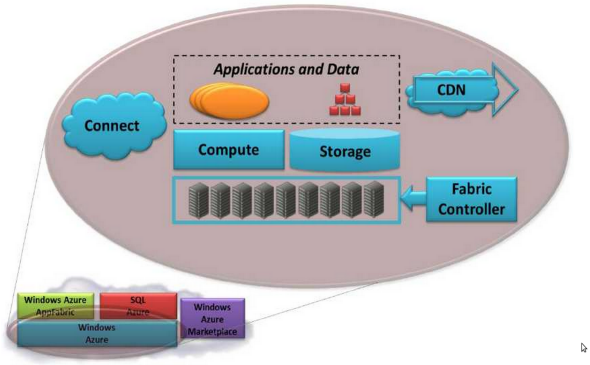


Figura 4.5: Componenti Azure tratta dal sito di Microsoft Azure

  - **Compute:** esegue applicazioni sui server della cloud. Le applicazioni possono essere sviluppati con la tecnologia .NET con l'ausilio dell'ambiente di sviluppo Visual Studio, oppure sviluppate con altri linguaggi come C++, Java, Ruby, PHP e Python con i tools che si preferiscono.
  - **Storage:** permette la memorizzazione di Blob, fornisce code per la comunicazione tra applicazioni e un semplice linguaggio di query. Le risorse sono accessibili tramite l'approccio REST[14].
  - **Fabric controller:** come suggerisce la figura 4.5, Windows Azure è eseguito su molteplici macchine. Il lavoro del Fabric Controller è di creare una rete tra le macchine in modo da creare maggior coesione e rendere possibile la creazione degl'altri servizi.
  - **Content delivery network (CDN):** è un servizio di caching, che migliora l'accesso frequente alle risorse.
  - **Connect:** permette il settaggio di politiche di sicurezza della rete come una cloud Privata.

[14]http://it.wikipedia.org/wiki/Representational_State_Transfer

- **SQL Azure:** servizio di database relazionale basato su server SQL in figura 4.6

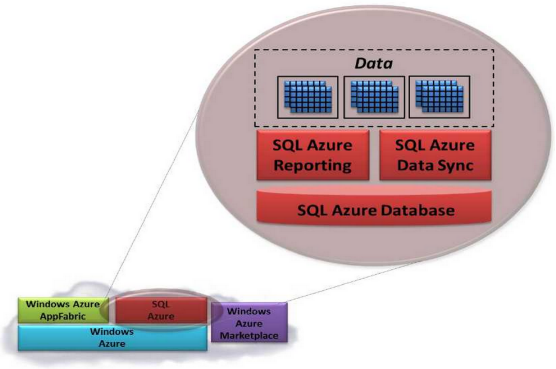


Figura 4.6: SQL Azure tratta dal sito di Microsoft Azure

  - **SQL Azure Database:** fornisce un DBMS cloud-based, convertendo il costo di aggiornamento degli Hard Disk (per ottimizzare le prestazioni) e licenze, in costo di utilizzo effettivo.
  - **SQL Azure Reporting:** versione di SQL Server Reporting Services (SSRS) per cloud, fornice report sulle operazioni del database.
  - **SQL Azure Data Sync:** sistema di replicazione per mantenere coerenti database situati in diversi datacenter

- **Windows Azure AppFabric:** servizi interni all'infrastruttura cloud in figura 4.7.

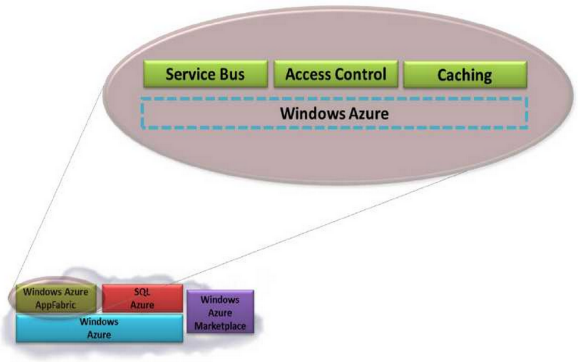


Figura 4.7: Azure AppFabric tratta dal sito di Microsoft Azure

- **Service Bus:** esporre un applicazione su internet è più difficile di quanto si pensi. Lo scopo di Service Bus è pertanto quello di semplificare questa procedure assegnando ad ogni applicazioni un URI[15], gestendo la traduzione degli indirizzi IP e relative regole di firewall.
- **Access Control:** semplifica la gestione degli accessi e dell'autenticazione fornendo un servizio per più applicazioni.
- **Caching:** sitema di caching per ottimizzare le operazioni molto frequenti.

- **Windows Azure Marketplace:** servizio online per la vendita di dati e servizi web in figura 4.8

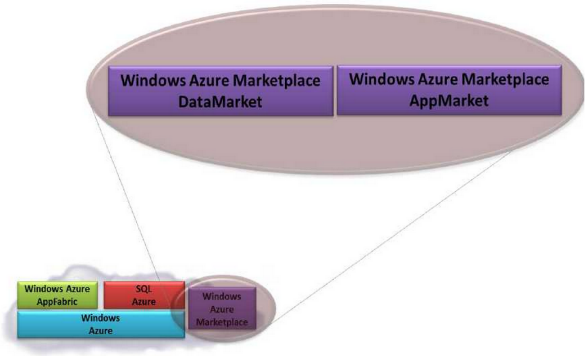


Figura 4.8: Marketplace tratta dal sito di Microsoft Azure

- **DataMarket:** nome in codice Dallas è un servizio di vendità di informazioni e dati. Nel sito di dallas troviamo informazioni e dati raccolti dalla Nasa[16] nelle spedizioni lunari sulla luna o i resoconti del livello di criminalità di ogni stato americano.
- **AppMarket:** permette ai creatori di servizi di avere visibilità e reperire clienti.

[15]http://it.wikipedia.org/wiki/Uniform_Resource_Identifier
[16]http://www.nasa.gov/

4.3 Amazon Elastic Compute Cloud

Amazon[17] pioniera dell'e-commerce alla fine degl'anni novanta, ha creduto anche nel cloud computing lanciando il suo servizio di IaaS, Amazon Elastic Compute Cloud (EC2)[18]. Dando la possibilità di godere dei benefici del cloud computing con la libertà di scelta dei propri strumenti di sviluppo, e un ampio controllo sulla sicurezza delle istanze, fornendo soluzioni pubbliche, private e ibride.

Per usare EC2 bastano le seguenti operazioni:

- Selezionare una delle diverse immagini (Sistema operativo e applicazioni) già pronte, o creare una amazon machine image (AMI) con il proprio sistema operativo, tools, librerie, e politiche di sicurezza.
- Configurare una rete tra le istanze
- Scegliere in quali datacenter caricare le istanze, se avere un IP statico o dinamico e se dotare le istanze di uno storage persistente.

Le caratteristiche principali di EC2 sono:

- **Amazon Elastic Block Store (EBS):** offre uno storage per le istanze indipendentemente dalla durata di vita delle istanze. Permette di creare volumi da 1GB a 1 TB e collegarli alle istanze, lo stesso volume può essere montato su un istanza alla volta. I volumi sono situati nelle Availability Zone e possono essere collegati solo alle istanze della stessa Availability Zone. Essendo volumi grezzi (raw) è possibile creare un file sistem o usarli come dispositivi a blocchi. Come ogni sistema di storage che si rispetti, possiede politiche di replicazione e snapshots nel Amazon S3[19]. Gli snapshots sono incrementali, ovvero a intervalli di tempo vengono salvati solo i blocchi modificati dopo l'ultimo snapshots.
- **Multi locazione:** per garantire continuità da fallimenti del data center e ottenere una miglior latenza, è possibile eseguire istanze in cinque data center: US East (Northern Virginia), US West (Northern California), EU (Ireland), Asia Pacific (Singapore), and Asia Pacific (Tokyo).
- **Indirizzo IP elastico:** è un indirizzo IP assegnato al proprio account, non alle singole istanze. Il possessore dell'account lo gestisce fino alla richiesta esplicita di rilascio. Questa soluzione è utile contro i fallimenti delle istanze o dei data center dove in tal caso si termina l'istanza guasta e si assegna l'indirizzo ad una nuova istanza.

[17] http://www.amazon.com/
[18] http://aws.amazon.com/ec2/
[19] http://aws.amazon.com/s3/

- **Amazon Virtual Private cloud (VPC):** permette di creare una rete sicura tra l'azienda e l'infrastruttura amazon, tramite una VPN dando la possibilità di configurare le politiche di sicurezza.

- **Amazon cloudWatch:** è lo strumento principale per il monitoraggio delle proprie applicaznioni. Permette la visualizzazione delle performance, utilizzo di risorse e fluttuazioni di richieste, nonchè statistiche e grafici.

- **Auto-scaling:** amazon da libera scelta sulle politiche di auto-scaling, dando la possibilità di ottimi livelli di performance (scale up) e riduzione dei costi (scale down). Lo scaling lavora con le metriche del cloudWatch o con uno scheduling definito dell'amministratore.

- **Elastic Load Balancing:** distribuisce automaticamente il traffico tra le istanze. Quando le richieste in arrivo su un istanza superano determinate soglie, il sistema le distribuisce su istanze con code di richieste più brevi.

- **High Performance computing (HPC) Clusters:** clienti con carichi di lavoro pesanti come applicazioni parallele o dipendenti dalle performance di rete, possono godere di un infrastruttura personalizzata alle loro necessità. É possibile creare cluster di CPU e di GPU (Graphics Processing Unit) per ottenere la potenza di elaborazione combinata di CPU e GPU con le performance di rete di una LAN. Questa soluzione è creata ad hoc per la ricerca scientifica.

Tipologie di risorse e prezzi

La figura 4.9 mostra le tipologie delle istanze, per maggiori dettagli sulle caratteristiche di ogni istanza visitare il sito[20]. Da notare il differente prezzo tra istanze unix-like e windows, presumibilmente dovuto a costi di licenza e tecnici specializati.

Region: EU (Ireland)

	Linux/UNIX Usage	Windows Usage
Standard On-Demand Instances		
Small (Default)	$0.095 per hour	$0.12 per hour
Large	$0.38 per hour	$0.48 per hour
Extra Large	$0.76 per hour	$0.96 per hour
Micro On-Demand Instances		
Micro	$0.025 per hour	$0.035 per hour
Hi-Memory On-Demand Instances		
Extra Large	$0.57 per hour	$0.62 per hour
Double Extra Large	$1.14 per hour	$1.24 per hour
Quadruple Extra Large	$2.28 per hour	$2.48 per hour
Hi-CPU On-Demand Instances		
Medium	$0.19 per hour	$0.29 per hour
Extra Large	$0.76 per hour	$1.16 per hour
Cluster Compute Instances		
Quadruple Extra Large	N/A*	N/A*
Cluster GPU Instances		
Quadruple Extra Large	N/A*	N/A*

* Cluster Compute and Cluster GPU Instances are currently only available in the US East (Virginia) Region.

Figura 4.9: Tariffario utilizzo risorse, tratto dal sito di amazon

[20] http://aws.amazon.com/ec2/pricing/

Capitolo 5

Sperimentazione

Come parte sperimentale di questa tesi, analizzeremo le performance di un applicazione su Google AppEngine, cercando di capire come reagirebbe ad un eventuale contratto di SLA.

5.1 Scenario

Lo scenario è quello di un applicazione di home trading. Un applicazione bancaria che permette l'acquisto e vendita di titoli online, che vuole garantire tempi brevi per le operazioni dei suoi clienti. Cosa succede se cinque minuti prima dell'apertura della borsa, milioni di clienti richiedono il proprio saldo? L'infrastruttura di Google AppEngine riuscirà a garantire i tempi stabiliti nel contratto di Service Level Agreement? L'applicazione scritta in Python con il framework webapp, consiste in un modulo html per il login. A login avvenuto, viene inviata una pagina con il saldo del cliente. Sempre con l'aiuto di Python, uno script con diversi threads simula l'operazione di saldo da parte di diversi utenti a intervalli di tempo. Dal momento che la prima operazione di una macchina virtuale appena istanziata ha un tempo di risposta maggiore, per via del caricamento dell'applicazione e relative librerie, le richieste in coda subiscono ritardo (figura 5.1).

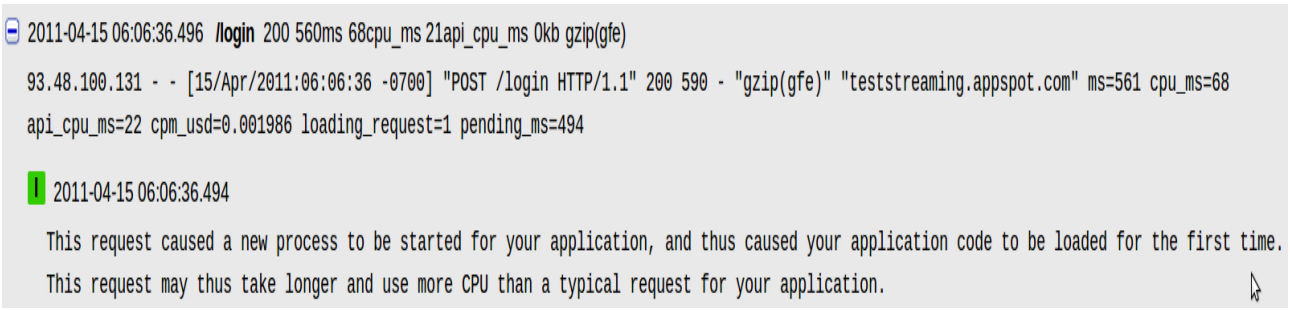


Figura 5.1: Prima richiesta tratta del pannello di controllo di AppEngine

5.2 Primo test

La strumentazione utilizzata per i seguenti test è composta da:

- processore intel i5 con 2,66 Ghz di frequenza
- 4 Gb di memoria ram
- sistema operativo Ubuntu[1] 10.10
- rete fastweb da 5 Mbps in download e 0,5 Mbps in upload

Al momento dei test sulla macchina non giravano applicazioni al di fuori del test Python, in modo da garantire una miglior realizzazione dei test. Per determinare il tempo massimo che dovrà rispettare l'operazione di saldo, in maniera empirica consideriamo la latenza media di 60 richieste lanciate a intervalli di 1 secondo e soddisfatte da 1 sola istanza. Il valore di latenza media in figura 5.2 è errato, in quanto calcolato all'aggiornamento della pagina mostrata in figura, sulla latenza delle richieste degl'ultimi 60 secondi. Dal momento che la pagina in figura e caricata qualche secondo dopo la fine del test, non vengono conteggiate le prime richieste. Per un accurato calcolo della media bisogna calcolarla con i dati forniti nei log. La media effettiva ottenuta dai log è di 43 ms.

[1] http://www.ubuntu-it.org/

Total number of instances	Average QPS*	Average Latency*	Average Memory
1 total	0.950	29.9 ms	6.9 MBytes

Instances

QPS*	Latency*	Requests	Errors	Age	Memory	Availability
0.950	29.9 ms	60	0	0:01:00	6.9 MBytes	Dynamic

* QPS and latency values are an average over the last minute.

Figura 5.2: Informazioni relative all'istanza utilizzata nel primo test tratta del pannello di controllo di AppEngine

Dopo aver lanciato il primo test (60 richieste a distanza di un secondo) in figura 5.3 possiamo vedere il comportamento del test. Come detto prima la prima richiesta di una nuova istanza richiede più tempo rispetto alle successive, ma nel grafico notiamo altre richieste con tempi sopra la media. É ragionevole dedurre che siano dovute a operazioni di monitoraggio della coda di richieste, per verificare se ci sia la necessità di aggiungere nuove istanze per far fronte alle richieste in arrivo. Sull'asse delle ascisse troviamo la durata del test (60 secondi), e su quello delle ordinate troviamo la latenza della risposta, ovvero dal tempo di arrivo della richiesta all'invio della risposta calcolata. La retta in verde rappresenta la latenza media (43 ms), tutte le risposte sopra la media si devono considerare come violazione del contratto di SLA.

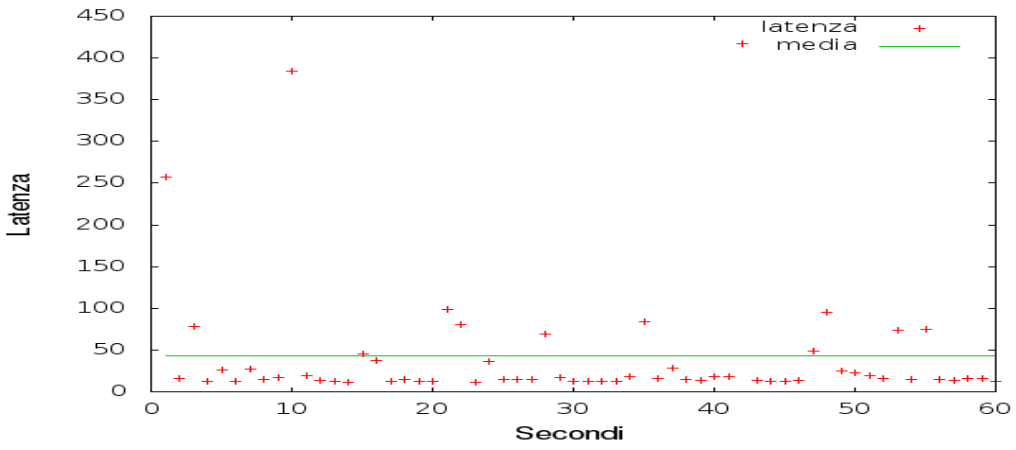


Figura 5.3: Risultati primo test

5.3 Secondo test

Trovato un tempo da attibuire all'operazione di saldo con il primo test, vogliamo vedere i risultati con picchi di richieste crescenti. Come mostra la figura 5.4 le richieste non arrivano più una al secondo, ma crescono in maniera esponenziale in base alla potenza di due, per simulare un flusso di richieste più realistico. Sull'asse delle ascisse troviamo la durata del test (7 secondi), e sull'asse delle ordinate troviamo il numero di richieste spedite al secondo. Il test è in esecuzione per 7 secondi con un massimo di 64 richieste al secondo.

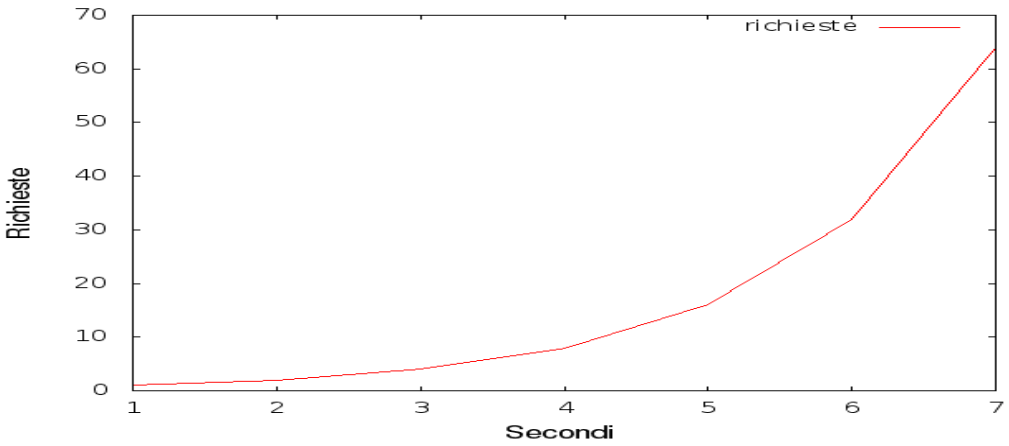


Figura 5.4: Andamento richieste secondo test

La figura 5.5 e la figura 5.6 mostrano i risultati del secondo test che è stato soddisfatto da 4 istanze. Anche in questo caso la latenza media in figura è errata. Calcolandola dai log, abbiamo una latenza media di 56 ms. Come possiamo vedere nel caso di richieste che crescono esponenzialmente si ha un degrado delle prestazioni, anche se si ha una distribuzione della richieste su più istanze.

Total number of instances	Average QPS*	Average Latency*	Average Memory
4 total	0.579	23.9 ms	6.8 MBytes

Instances

QPS*	Latency*	Requests	Errors	Age	Memory	Availability
0.700	23.3 ms	39	0	0:00:04	6.8 MBytes	Dynamic
0.800	24.6 ms	45	0	0:00:08	6.8 MBytes	Dynamic
0.083	42.0 ms	2	0	0:00:01	6.7 MBytes	Dynamic
0.733	21.8 ms	41	0	0:00:07	6.8 MBytes	Dynamic

* QPS and latency values are an average over the last minute.

Figura 5.5: Informazioni relative all'istanza utilizzata nel secondo test tratta del pannello di controllo di AppEngine

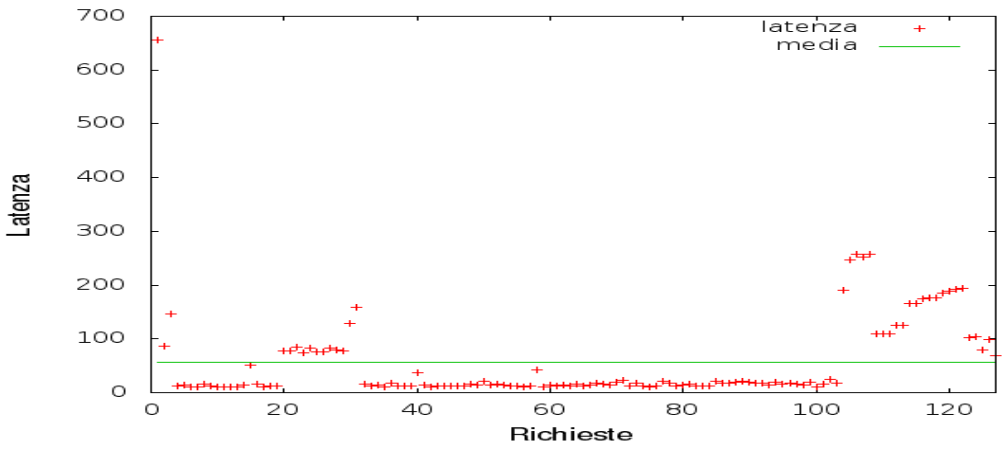


Figura 5.6: Risultati secondo test

Come spiegato nel corso di questa tesi, le politiche di scale-up permettono di aggiungere istanze per far fronte ad aumenti di richieste. Mentre le politiche di scale-down permettono di deallocare istanze quando non è giustificato più il loro utilizzo, ad esempio un basso rapporto di richieste/istanze. Lo scale-down permette quindi di minimizzare il costo delle istanze da parte dell'utente. Dal momento che parliamo di contratti SLA, gli algoritmi di scale devono essere un trade-off tra costi e prestazioni. Per quanto riguarda lo scale-down, GAE tiene in memoria le istanze utilizzate per circa 40 minuti dopo il loro utilizzo facendo pagare solo il tempo effettivo di utilizzo. La figura 5.7 mostra invece il comportamento dell'algoritmo di scale-up nel secondo test.

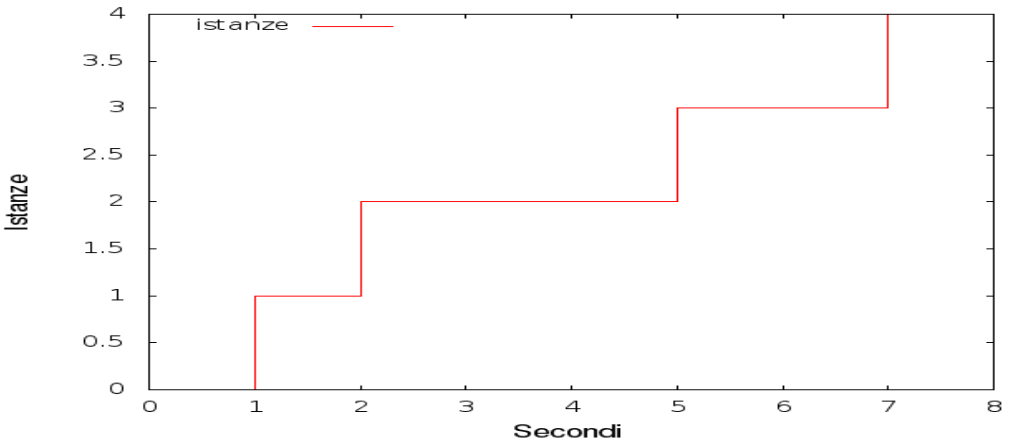


Figura 5.7: Scale-up

5.4 Terzo test

Nel terzo test vogliamo vedere i risultati con picchi di richieste con nessuna istanza allocata. Impostiamo il nostro script per effettuare 64 richieste in un secondo (picco finale del secondo test) con zero istanze allocate inizialmente. In figura 5.8 vediamo che il nostro test è stato eseguito da 6 istanze, con una latenza media di 529 ms in figura 5.9.

Total number of instances	Average QPS*	Average Latency*	Average Memory
6 total	0.228	31.3 ms	6.7 MBytes

Instances

QPS*	Latency*	Requests	Errors	Age	Memory	Availability
0.217	20.8 ms	10	0	0:00:14	6.7 MBytes	Dynamic
0.150	23.3 ms	6	0	0:00:14	6.6 MBytes	Dynamic
0.317	24.1 ms	16	0	0:00:14	6.7 MBytes	Dynamic
0.067	62.0 ms	1	0	0:00:14	6.6 MBytes	Dynamic
0.300	17.3 ms	15	0	0:00:14	6.7 MBytes	Dynamic
0.317	56.0 ms	16	0	0:00:14	6.8 MBytes	Dynamic

* QPS and latency values are an average over the last minute.

Figura 5.8: Istanze terzo test

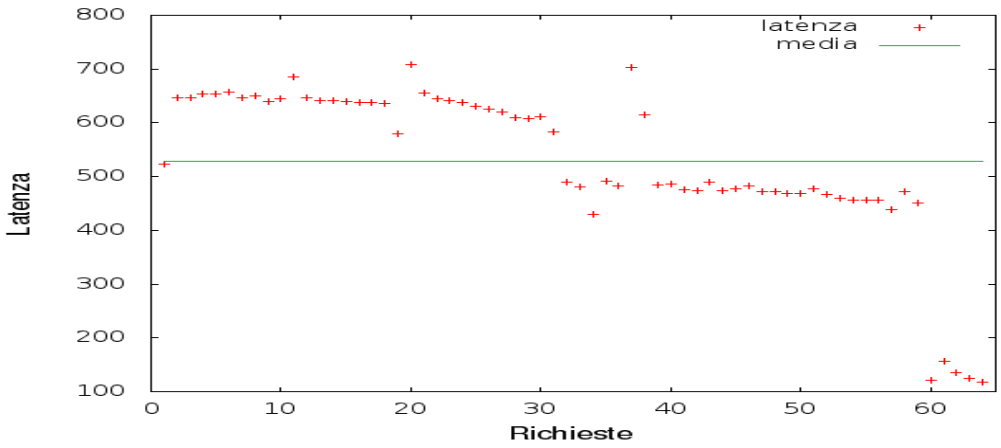


Figura 5.9: Risultati terzo test

5.5 Considerazioni

Ipotizzando che il tempo dell'operazione di login in un ipotetico contratto di SLA sia 43 ms, notiamo che i nostri test hanno latenze superiori. Nel secondo test si ha una latenza media di 56 ms con il 31% di richieste oltre il limite. Dal momento che le richieste crescono gradualmente, seppure in maniera esponenziale, le istanze allocate per le richieste precedenti sono già pronte per soddisfare le richieste successive, facendo risparmiare gli onerosi tempi di allocazione. Nel terzo test si ha una latenza media di 529 ms con il 100% di richieste oltre il limite. Non essendoci nessuna istanza già allocata, si paga un forte prezzo in allocazione. Le 64 richieste al secondo vengono soddisfatte da 6 istanze mentre nel secondo test all'arrivo delle 64 richieste c'erano già 3 istanze allocate. Alla luce di questi test possiamo affermare che le performace degradano fortemente in base alla rapidità dell'aumento delle richieste. Il tempo di allocazione fa crescere le code delle richieste sommando il tempo di allocazione a tutte le richieste in coda. Un margine di miglioramento delle performance può avvenire nei 40 minuti successivi ad un picco di richieste, dove le istanze allocate restano in memoria riducendo i tempi di allocazione.

Capitolo 6

Conclusioni

Nel corso di questa tesi abbiamo illustrato che cos'è il cloud computing, spiegando come si è giunti alla sua invenzione grazie all'unione del concetto di utility computing di John McCarthy e alle tecnologie di grid computing e virtualizzazione. Abbiamo illustrato le tecniche di realizzazione dei moderni data center, che permettono l'utilizzo del cloud computing da parte dei SaaS provider e SaaS user pagando solo le risorse che effettivamente utilizzano, stravolgendo i vecchi concetti di hosting. Essendo il cloud computing un settore relativamente giovane, sono ancora molte le sfide alla ricerca non ancora superate, il risparmio energetico e l'allocazione automatica sono solo alcune delle più importanti sfide ancora aperte in ambito scientifico. Come argomento principale di questa tesi abbiamo analizzato la garanzia del servizio che il cloud computing offre, soffermandoci sugli strumenti offerti in ambito di stipulazione di contratti di Service Level Agreement, mettendo a confronto i due maggiori framework che permettono l'incontro della domanda e dell'offerta e la possibilità di esprimere i propri requisiti di qualità in maniera formale. In seguito abbiamo analizzato quello che propone il mercato del cloud computing ai giorni nostri, mettendo a confronto le tre maggiori soluzioni attualmente sul mercato, che purtroppo visto lo stato dell'arte attuale non offrono la possibilità di contrattazioni sulla qualità del servizio, limitandosi ad impegnarsi nei limiti del possibile per garantire il miglio servizio possibile. Infine abbiamo eseguito alcuni test sperimentali sul servizio di di cloud computing offerto da Google, per valutare in maniera empirica la reale qualità del servizio, che attualmente siamo in grado di ottenere. Dai risultati finali dei test emerge che al momento la qualità del servizio lascia molto a desiderare. Nello specifico si hanno degradazioni delle performance al verificarsi dei cosidetti flash crowd[1], fenomeno che proprio il cloud computing si propone di eliminare grazie alla possibilità di allocare risorse in base alle richieste. In conclusione il cloud computing è la soluzione ottimale per favorire la migrazione delle tipiche applicazioni Desktop verso il web, ma non garantisce la qualità del servizio che molte applicazioni necessitano.

[1] ondate di persone

Bibliografia

[1] Ramnath K. Chellappa - *Intermediaries in cloud-computing: A new computing paradigm*, INFORMS Dallas, Dallas (TX), 26-29 Ottobre 1997

[2] Ian Foster, Yong Zhao, Ioan Raicu, Shiyong Lu - *Cloud Computing and Grid Computing 360-Degree Compared*, IEEE Grid Computing Environments Workshop (GCE'08), Austin (TX), 12-16 Novembre 2008, pp. 1-10

[3] Ian Foster Carl Kesselman - *The Grid: Blueprint for a New computing Infrastructure*, 1999 pubblicato da Morgan Kaufmann Publishers, 675 pagine

[4] Antimo Musone - *Windows Azure Platform*, Academic Tour 2010, Bologna, 29 Ottobre 2010

[5] Michael Armbrust e altri - *Above the clouds: A Berkeley View of cloud computing*, Electrical Engineering and Computer Sciences, University of California, Berkeley (CA), 10 Febbraio 2009

[6] Qi Zhang ed altri - *cloud computing: state-of-the-art and research challenges*, Journal of Internet Services and Applications, Vol. 1, No. 1, Maggio 2010, pp. 7—18

[7] Erik van der Kouwe - *Virtual machines*, 2006 Bachelor Thesis for Computer Science

[8] Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung - *The Google File System*, 19th ACM Symposium on Operating Systems Principles (SOSP'03), Lake George (NY), 19-22 Ottobre 2003

[9] Fay Chang. ed altri - *Bigtable: A Distributed Storage System for Structured Data*, 7th Symposium on Operating System Design and Implementation (OSDI'06), Seattle (WA), 6-8 Novembre 2006

[10] Daniel Nurmi ed altri - *The Eucalyptus Open-source cloud-computing System*, 9th IEEE/ACM International Symposium on Cluster computing and the Grid at IEEE Computer Society (CCGRID'09), Washington (WA), 18-21 Maggio 2009,Vol. 0, pp. 124-131

[11] Renzo Davoli - *VDE: Virtual Distributed Ethernet*, First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities (TRIDENTCOM'05), Trento, 23-25 Febbraio 2005, pp.213-220

[12] Bhuvan Urgaonkar e altri - *Dynamic provisioning of multi-tier Internet applications*, 2nd International Conference on Autonomic computing (ICAC'05), Seattle (WA) , 13-16 Giugno 2005, pp.217-228

[13] Qi Zhang, Ludmila Cherkasova, Evgenia Smirni - *A regression-based analytic model for dynamic resource provisioning of multi-tier applications*, 4th International Conference on Autonomic Computing (ICAC'07), Jacksonville (FL), 11-15 Giugno 2007, pp. 27

[14] Evangelia Kalyvianaki, Themistoklis Charalambous, Steven Hand - *Self-adaptive and self-configured CPU resource provisioning for virtualized servers using Kalman filters*, 6th International Conference on Autonomic Computing (ICAC'09), Barcelona, 15-19 Giugno 2009

[15] Peter Bodík ed altri- *Statistical machine learning makes automatic control practical for Internet datacenters*, Workshop on Hot Topics in cloud computing (Hotcloud'09), San Diego (CA), 15 Giugno 2009

[16] Christopher Clark ed altri - *Live migration of virtual machines*, 2nd conference on Symposium on Networked Systems Design & Implementation (NSDI'05), Berkeley (CA), 2-4 Maggio 2005

[17] Chandra Chekuri, Sanjeev Khann - *On multi-dimensional packing problems*, Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'99) Philadelphia (PA), 17-19 Gennaio 1999

[18] Bo Li ed altri - *Enacloud: An Energy-Saving Application Live Placement Approach for cloud computing Environments*, IEEE 2009 International Conference on Cloud Computing (CLOUD-II 2009), 21-25 Settembre 2009, Bangalore (India), pp.17-24

[19] Shekhar Srikantaiah, Aman Kansal, Feng Zhao - *Energy aware consolidation for cloud computing*, Workshop on Power Aware Computing and Systems (HotPower '08), San Diego (CA), 7 Dicembre 2008

[20] James Hamilton - *Cooperative expendable micro-slice servers (CEMS): low cost, low power servers for Internet-scale services*, Conference on Innovative Data Systems Research (CIDR '09), Pacific Grove (CA), 4-7 Gennaio 2009

[21] David M. Brooks ed altri - *Power-aware microarchitecture: design and modeling challenges for the next-generation microprocessors*, IEEE Micro, Novembre-Dicembre 200, Vol. 20, No. 6, pp. 26-44

[22] Nedeljko Vasic, Martin Barisits, Dejan Kostić - *Making cluster applications energy-aware*, 1st workshop on Automated Control for Datacenters and Clouds (ACDC '09), Barcellona, 19 Giugno 2009

[23] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Pat - *VL2: a scalable and flexible data center network*, SIGCOMM'09, Barcellona, 17-21 Agosto 2009

[24] Jeffrey Dean, Sanjay Ghemawat - *MapReduce: simplified data processing on large clusters*, 6th Symposium on Operating System Design and Implementation (OSDI'04), San Francisco (CA), 5 Dicembre 2004

[25] Nuno Santos, Krishna P. Gummadi, Rodrigo Rodrigue - *Towards trusted cloud computing*, Workshop on Hot Topics in cloud computing (Hotcloud'09), San Diego (CA), 15 Giugno 2009

[26] Karthik Kambatla, Abhinav Pathak, Himabindu Puch - *Towards optimizing Hadoop provisioning in the cloud*, Workshop on Hot Topics in cloud computing (Hotcloud'09), San Diego (CA), 15 Giugno 2009

[27] Thomas Sandholm, Kevin Lai - *MapReduce optimization using regulated dynamic prioritization*, 11th international joint conference on Measurement and modeling of computer systems (SIGMETRICS'09), Seattle (WA), 15-19 Giugno 2009

[28] Matei Zaharia ed altri - *Improving MapReduce performance in heterogeneous environments*, 8th Symposium on Operating System Design and Implementation (OSDI'08), San Diego (CA), 8-10 Dicembre 2008

[29] Rajagopal Ananthanarayanan ed altri - *Cloud analytics: do we really need to reinvent the storage stack?*, Workshop on Hot Topics in cloud computing (Hotcloud'09), San Diego (CA), 15 Giugno 2009

[30] Swapnil Patil ed altri - *In search of an API for scalable file systems: under the table or above it?*, Workshop on Hot Topics in cloud computing (Hotcloud'09), San Diego (CA), 15 Giugno 2009

[31] Ken Church, Albert Greenberg, James Hamilto - *On delivering embarrassingly distributed cloud services*, 7th Workshop on Hot Topics in Networks (HotNets-VII), Calgary (Alberta), 6-7 Ottobre 2008

[32] Vytautas Valancius, Nikolaos Laoutaris, Laurent Massouli - *Greening the Internet with nano data centers*, 5th international conference on Emerging networking experiments and technologies (CoNEXT'09), Roma, 1-4 Dicembre 2009

[33] Abhishek Chandra, Jon Weissma - *Nebulas: using distributed voluntary resources to build clouds*, Workshop on Hot Topics in cloud computing (Hotcloud'09), San Diego (CA), 15 Giugno 2009

[34] Alessandro Raffio, Christina Tzivisko - *Service Level Agreement Languages* http://home.dei.polimi.it/ghezzi/_PRIVATE/SLA-final.pdf

[35] Tushar Deepak Chandra ed altri - *Paxos Made Live - An Engineering Perspective*, 26th Symposium on Principles of Distributed Computing (PODC'07), Portland (OR), 12-15 Agosto 2007