PCD Module 5.4

Ingegneria dei sistemi distribuiti: il Cloud

Introduzione

CC è una delle scelte principali nella progettazione di applicazioni distribuite su larga scala, grazie alla scalabilità, apertura e eterogeneità, disponibilità.

Cloud computing

Si spostano i servizi, per questioni di costi e vantaggi, fuori in strutture centralizzate, interne o esterne, trasparenti alla locazione. Driver principali: Google, Yahoo, Amazon, Microsoft...

Rendendo i dati disponibili sul cloud, possono essere acceduti più facilmente e ubiquamente ad un costo minore e con più opportunità di collaborazione, integrazione e analisi su una piattaforma comune condivisa. Ma ci sono anche problematiche di sicurezza e performance.

Da CapEx a OpEx

In passato, per sviluppare un servizio ci voleva una grande spesa capitale (CapEx) per costruire l'infrastruttura che reggesse una domanda alta.

Il CC permette di spendere tracciando le risorse e spostandosi verso OpEx, spese operative, permettendo allo stesso tempo scalabilità e flessibilità.

Grazie al CC è possibile sviluppare progetti da zero senza dover pensare a un'infrastruttura server che potrebbe poi non servire.

Background storico

.....

La rete è il computer

Dalla rete al web

Da statico a dinamico

Dai cluster al cloud

Modelli di servizio

- **Client cloud**: browser, emulatore, app mobile, ecc.

che si interfaccia con:

- **Software as a Service**: servizi, Google Apps, salesforce, WebEx...
- **Platform as a Service**: fondamenta, Google App Engine, Azure, Heroku...
- **Infrastructure as a Service**: infrastruttura, Amazon EC2, Google Computing Engine, Open Stack

Cosa posso controllare

OPE = On premise environment

```
                        OPE     IaaS    PaaS    SaaS
Applicazioni            x       x       x
Dati                    x       x       x
Runtime             x       x
Middleware          x       x
OS                      x
Virtualizzazione        x
Server                  x
Storage             x
Networking          x
```

Tecnologie sfruttabili

- Storage poco costoso
- CPU con larghezza di banda tale da supportare computazioni importanti poco costoso
- Algoritmi e tecnologie client sofisticati (HTML, CSS, AJAX, REST)
- Client broadband
- SOA
- Implementazioni infrastrutturali dei provider
- Virtualizzazione commerciale

Modelli di fornitura

Pubblico

Quando i servizi sono forniti su una rete aperta all'uso pubblico. Ci sono poche differenze col cloud privato,

a parte le misure di sicurezza. Generalmente, i provider possiedono e gestiscono le infrastrutture e le rendono disponibili tramite Internet (senza connettività diretta). Es. Google, Amazon AWS, Microsoft, ecc...

Privato

Infrastruttura operata solo per una singola organizzazione, gestita internamento o da una terza parte, ospitata internamente o esternamente.

Creare un cloud privato richiede un livello significativo e grado di impegno nel virtualizzare l'ambiente aziendale, e richiede che le aziende rivalutino decisioni riguardanti risorse esistenti. Se fatto bene può migliorare il business, ma ogni passo comporta problematiche di sicurezza che devono essere prese in considerazione. I datacenter proprietari sono generalmente costosi.

Questi asset devono essere aggiornati periodicamente --> spese addizionali. Critiche forti poichè non ci sono dei gran vantaggi.

Comunità

Condivide l'infrastruttura tra diverse organizzazioni con uno specifico scopo (es. sicurezza, giurisdizione, ecc..), gestita internamente o da una terza parte e ospitata internamente o esternamente. I costi sono spalmati su un numero di utenti minore rispetto a un cloud pubblico, ma più di un cloud privato.

Ibrido

Composizione di due o più cloud (private/pubbliche/community, da provider diversi) che rimangono entità distinte ma sono collegate assieme, offrendo i benefici di modelli di produzione multipli.

Caratteristiche e funzionalità

- **Agilità** migliora con l'abilità dell'utente di rieseguire provisioning di risorse tecnologiche infrastrutturali.
- **Accesso alle API** da parte del software, permettendo alle macchine di interagire con il software cloud nello stesso modo in cui un'interfaccia utente tradizionale facilita l'interazione tra umani e computer. I sistemi cloud tipicamente usano API basate su **REST**.
- **Costo**: i provider dicono che il cloud riduca i costi. Il modello di fornitura di una cloud pubblica converte le spese capitali in spese operazionali. Ciò abbassa gli ostacoli per usufruire di tali servizi, poichè l'infrastruttura è fornita da terze parti e non ha bisogno di essere acquistata per un uso infrequente.
- **Indipendenza dei dispositivi e delle locazioni**: permette agli utenti di accedere al sistema tramite un browser, non ha importanza la posizione o lo strumento che usano.
- **Tecnologie di virtualizzazione**: permettono la condivisione dei server e dello storage. Le applicazioni possono essere migrate facilmente da un server fisico ad un altro.
- **Multilocazione**: permette la condivisione di risorse e costi su un largo numero di utenti, permettendo così la centralizzazione di infrastrutture con costi minori (es. affitto), la capacità di carico a picco

aumenta, miglioramento dell'utilizzo e dell'efficienza per sistemi usati al 10-20%.
- **Affidabilità** migliora con l'uso di siti multipli ridondati, rendendo un cloud ben progettato adatto alla continuità e al recupero dai disastri.
- **Scalabilità** ed elasticità tramite fornitura dinamica on-demand di risorse fornite in tempo reale all'utente
- **Perfomance** monitorate, architetture poco accoppiate e consistenti costruite utilizzando i web service come interfaccia di sistema.
- **Sicurezza**: può migliorare grazie alla centralizzazione, ma ci sono ancora dei dubbi sulla perdita di controllo su certi dati sensitivi, e la mancanza di sicurezza per stored kernel.
- **Manutenzione** di sistemi cloud è facile, perchè non hanno bisogno di essere installati su ogni computer e possono essere acceduti da posti diversi.

Caratteristiche essenziali (NIST)

- **Self-service on daemand**: un consumatore può unilateralmente fornire capacità di calcolo, ad esempio tempo server e storage in rete, come necessario automaticamente senza richiedere l'interazione umana con ogni fornitore.
- **Accesso vasto alla rete**: le funzionalità sono disponibili sulla rete e accessibili tramite meccanismi standard che favoriscono l'eterogeneità.
- **Pooling di risorse**: le risorse del provider sono condivise per servire utenti multipli usando un modello multilocato, con risorse differenti fisiche e virtuali assegnate dinamicamente e riassegnate in accordo con le richieste del consumatore.
- **Elascitià rapida**: le funzionalità possono essere fornite e rilasciate elasticamente, in certi casi automaticamente, per scalare rapidamente in grande e in piccolo in base alla domanda. Al consumatore le risorse appaiono illimitate e possono essere regolate in qualsiasi momento.
- **Servizio misurato**: i sistemi cloud controllano e ottimizzano automaticamente l'uso delle risorse sfruttando capacità di conteggio ad un certo livello di astrazione appropriato per il tipio di servizi. L'uso di risorse può essere monitorato, controllato e segnalato, fornendo trasparenza sia al provider che al consumatore.

SaaS

Gli utenti hanno accesso ai software applicativi e al database, disponibili solo online - via browser o app - come servizio. I fornitori gestiscono l'infrastruttura. Riferito come "software on demand" e con prezzi pay-per-use o tramite abbonamenti.

Nel modello SaaS, i fornitori cloud installano e operano software applicativo nel cloud, e gli utenti cloud accedono al software dai client cloud. Gli utenti cloud non gestiscono l'infrastruttura e e la piattaforma su cui l'applicazione viene eseguita: ciò elimina il bisogno di installare e eseguire applicazioni nel computer dell'utente, semplificando manutenzione e supporto.

Le applicazioni cloud sono diverse da altre applicazioni in termini di scalabilità, che può essere ottenuta

clonando task su multiple macchine virtuali a tempo di esecuzione per incontrare la domanda di cambio lavoro.

Per accomodare un numero maggiore di utenti cloud, le applicazioni possono essere multilocate: ogni macchina serve più di un'organizzazione ed è comune riferirsi a tipi speciali di applicazioni cloud con nomi simili: desktop-aaS, business proces-aaS, test-aaS...

Il modello di prezzo per applicazioni SaaS è tipicamente una quota mensile o annuale per utente, così il prezzo può essere scalabile e aggiustabile se gli utenti vengono aggiunti o rimossi in qualsiasi momento.

Pro e contro

- **Pro**: SaaS permette alle aziende la riduzione dei costi operativi IT, mandando la manutenzione HW e SW in outsourcing, così da riallocare i costi relativi a gestione HW + SW + dipendenti. Con applicazioni ospitate centralmente, gli aggiornamenti possono essere rilasciati senza bisogno che gli utenti installino nuovo software.
- **Contro**: i dati degli utenti sono memorizzati sui server cloud del provider. Perciò, potrebbero esserci accessi non autorizzati ai dati. Per tale motivo, gli utenti stanno adottando sempre di più sistemi di gestione intelligenti delle chiavi di terze parti per mettere in sicurezza i propri dati.

IaaS

Nel modello più essenziale di servizio cloud, i provider di IaaS offrono macchine fisiche o virtuali e altre risorse come servizio. Un ipervisore (OpenStack, Xen, HyperV) viene eseguito sulla macchina virtuale come ospite. Insiemi di ipervisori all'interno del sistema di supporto operazionale del cloud possono supportare un numero elevato di macchine virtuali e l'abilità di scalare servizi in grande e in piccolo, in accordo con i requisiti variabili dei clienti.

Le cloud IaaS a volte possono offrire risorse aggiuntive come un disco virtuale, una libreria immagini, raw block storage, memorizzazione di file e oggetti, firewall, bilanciatori di carico, indirizzi IP, VLAN, bundle software.

I provider cloud IaaS forniscono queste risorse **on-demand** dai loro grandi pool installati nei datacenter. Per la connettività su larga scala, i clienti possono scegliere sia l'internet che il carrier clouds (VPN dedicate) ??

Questo paradigma è stato recentemente esteso a sensori e attuatori as a Service.

Per produrre le applicazioni, gli utenti cloud installano immagini degli OS e il software sull'infrastruttura cloud. In questo modello, gli utenti cloud aggiornano e mantengono l'OS e i software. I provider tipicamente fatturano i servizi IaaS su una base di utility utilizzate: quantità di risorse allocate e consumate.

PaaS

A questo livello di servizio, i venditori hanno cura dell'infrastruttura sottostante, dando agli sviluppatori solamente la piattaforma su cui ospitare la propria applicazione. Gli sviluppatori controllano l'applicazione, non come in SaaS. Gli sviluppatori non hanno a che fare con l'infrastruttura hardware e i problemi riguardanti OS e sistema (aggiornamenti, patch, ecc.).

Le app create su piattaforma PaaS sono applicazioni SaaS: gli utenti di tali app non hanno controllo sul codice. Un utente PaaS è uno sviluppatore di SaaS.

In comune tra le varie PaaS

- SOA, basato su REST
- Servizi disponibili fornite dalla piattaforma tramite API per
  - memorizzare e ritrovare dati
  - eseguire e gestire task a lungo termine
  - comunicazione
  - configurazione app e gestione

Esempi di PaaS

Vedi slide

Orleans Framework

Modello ad attori per organizzare le app cloud, chiamate grains. Basato su MS Azure.

Google App Engine, caso di studio

- Piattaforma e infrastruttura per sviluppo ed esecuzione di app cloud. Insieme di servizi ad alto livello accessibile tramite le App Engine API. Servizi GAE = Chiamate a sistema degli OS.
- Supporto a diversi linguaggi (Java e JVM-derivati, Python, Go) e piattaforme - **SDK open source**.
- Modello di sicurezza basato sul **sandboxing**: le applicazioni GAE vengono eseguite in un ambiente ristretto chiamato sandbox. Alcune azioni non sono permesse (es. apertura file locale per scrittura, apertura socket,...). In alternativa devono essere utilizzati i Servizi GAE
- Infrastruttura e runtime totalmente gestito da Google, il sistema si autoscala per allocare più istanze dell'app se richiesto. QoS negoziato al setup dell'account.
- Server di sviluppo fornito per testare il loro codice (con limitazioni) prima di entrare in produzione
- Strumenti di profilazione e amministrazione + servizi
- Progetti backend open source compatibili con GAE - AppScale, TyphoonAE, ...

Servizi principali

- Modello di interazione orientato ai servizi - REST style

- Categorie base
  - memorizzazione dati, recupero e ricerca
  - gestione dei processi e computazioni
  - comunicazione
  - configurazione e gestione app

Memorizzazione dati, recupero e ricerca

- **Google Cloud SQL**: servizio web completamente gestito che permette di creare, configurare e usare database relazionali ospitati sul Google Cloud
- **Datastore**: memorizzazione di dati per mezzo di oggetti senza schema, fornisce uno storage robusto e scalabile per le app, API ricche per la modellazione dei dati, linguaggio di query SQL-like chiamato GQL.
- **Blobstore**: permette alle applicazioni di fornire oggetti dati pesanti (video, immagini, ...) che sono troppo grandi per essere memorizzati in Datastore.
- **Memcache**: memoria cache distribuita e in memoria che può essere usata per migliorare le performance.
- **Dedicated Memcache**: fornisce una capacità di cache fissa assegnata esclusivamente per l'applicazione.
- **Logs**: fornisce accesso programmatico ai log dell'app e richieste.
- **Search**: permette all'applicazione di eseguire ricerche simil-Google su dati strutturati come testi, HTML, atom, numeri, date, geolocation.
- **HRD Migration Tool**: migra i dati delle applicazioni ospitate nel vecchio Datastore/Blobstore al nuovo Datastore.
- **Google Cloud Storage Client Library**: permette all'applicazione di leggere file da e scrivere file sui bucket del Google Cloud Storage, con gestione interna degli errori.

Comunicazione

- **Channel**: crea una connessione persistente tra l'app e i server Google, così da poter inviare messaggi a client Javascript in real time senza polling.
- **Google Cloud Endpoints**: permette la generazione automatica di API, rendendo più facile creare un backend web per client web e mobile.
- **Mail**: invia messaggi a amministratori e utenti con account Google, e riceve mail da diversi indirizzi.
- **URL Fetch**: usa l'infrastruttura di rete di Google per fornire richieste a URL web HTTP e HTTPS.
- **XMPP**: permette ad un'applicazione di inviare e ricevere messaggi in chat verso e da ogni servizio di messaggistica compatibile con XMPP.
- **Sockets**: permette di supportare sockets in uscita utilizzando le librerie specifiche per il linguaggio built-in.

Gestione dei processi e computazioni

- **Task Queue**: permette alle applicazioni di svolgere lavoro fuori da una richiesta dell'utente, e organizzare tale lavoro in piccole unità discrete chiamate "task", da esegurie in un secondo momento.
- **Scheduled Tasks**: permette alle applicazioni di configurare task regolarmente programmati per operare ad un certo orario o ad intervalli regolari.
- **Images**: manipola, combina, migliora immagini, converte in diversi formati, permette di chiedere i metadati di un'immagine (es. dimensione o frequenza colori).
- **MapReduce**: adattamento ottimizzato del modello di computazione MapReduce per computazioni distribuite efficiente su dataset larghi.

Configurazione e gestione app

- **App Identity**: fornisce accesso nel codice all'identità dell'applicazione, fornisce un framework per verificare l'identità tramite OAuth.
- **Capabilities**: fornisce la rilevazione delle interruzioni e manutenzione programmata per API specifiche e servizi, in modo che l'applicazione può escluderli o informare gli utenti.
- **SSL for Custom Domains**: permette alle applicazioni di essere servite con HTTPS e HTTP tramite un dominio personalizzato (invece di appspot.com).
- **Users**: permette alle applicazioni di loggarsi tramite Google Accounts o OpenID, e di memorizzare gli utenti tramite UUID
- **Modules**: permette ai dev di fattorizzare grandi app in componenti logici che condividono servizi stateful e comunicano in modo sicuro.
- **Multitenancy**: rende facile la compartimentazione dei dati per servire più organizzazioni client per una singola istanza dell'applicazione
- **Remote**: permette ad app esterne di accedere ai servizi App Engine in modo trasparente.
- **Traffic Splitting**: permette di lanciare nuove funzionalità in modo graduale (rollout), anche tramite AB Testing.

Dinamica delle richieste

- Ogni richiesta è prima ispezionata da un frontend di App Engine. Ce ne sono più di una e c'è un bilancio di carico che gestisce l'allocazione di richieste alle macchine
- Il frontend spedisce la richiesta al relativo handler dell'applicazione (tramite nome a dominio). L'obiettivo può essere un file statico o un handler.
- Le richieste gestite dall'handler sono reindirizzate ai server dell'app. Un server specifico viene selezionato e un'istanza dell'applicazione avviata. Se c'è già un'istanza in esecuzione può essere riutilizzata.
- L'handler della richiesta appropriato viene invocato e genera una risposta per la richiesta, basata sul codice dell'app
- Strategie di bilanciamento del carico e distribuzione delle richieste di applicazione hanno lo scopo di garantire un throughput elevato di richieste, con handler veloci. Quante istanze debbano essere avviate e quante richieste distribuite dipende dal traffico dell'applicazione e dall'uso di pattern.
- L'applicazione esegue il codice in un ambiente sandboxed

- Il server attende affinchè l'handler termina e ritorna la risposta al frontend, completando la richiesta
- Il frontend costruisce la risposta finale al client

Osservazioni

- Non viene garantito che due richieste vengano eseguite sulla stessa macchina, anche se le richieste arrivano una dopo l'altra dallo stesso client.
- Istanze multiple di app diverse possono essere eseguite sulla stessa macchina senza che una dia fastidio all'altra.

Ambiente runtime

- La sandbox è un ambiente isolato, basato su Python, Java o Go.
- Sandboxing è una scelta chiave: previene alle app di eseguire codice malevolo e permette ad App Engine di eseguire un bilanciamento automatico dei carichi di lavoro
- Restrizioni: nessun accesso diretto all'OS, uso diretto dei thread non permesso.