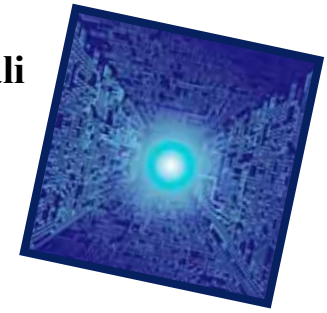
Octopus: una cloud self-service di machine virtuali



Antonio Cisternino, Maurizio Davini, Marco Mura

IT Center Università di Pisa

Abstract. Alla base delle infrastrutture Cloud si trovano i sistemi di virtualizzazione: l'allocazione dinamica delle risorse di un cloud richiede una flessibilità che le architetture tradizionali, con un sistema operativo in esecuzione direttamente su una macchina fisica, non sono in grado di fornire. In questo articolo presentiamo Octopus, un gestore di una rete di hypervisor, caratterizzato da un modello basato sull'idea di self-service via Web. Per gestire e coordinare un numero sempre crescente di macchine virtuali, secondo una politica di allocazione delle risorse, Octopus usa il sistema esperto CLIPS per definire in modo generale le regole che ne governano il comportamento.

1. Introduzione

Uno dei cardini del Cloud computing [1] è indubbiamente la capacità di virtualizzare le risorse in modo da favorire un disaccoppiamento tra l'erogazione di servizi e l'infrastruttura IT che li offre; le tecnologie di virtualizzazione svolgono quindi un ruolo importante nella realizzazione di Cloud pubbliche e private, e le macchine virtuali sono rapidamente divenute l'unità di allocazione di risorse virtuali alla base del cosiddetto modello IaaS (*Infrastructure as a Service*), a sua volta considerato alla base di modelli più articolati come PaaS (*Platform as a Service*) e SaaS (*Software as a Service*).

Le macchine virtuali sembrano offrire un'interfaccia ideale tra chi vuole offrire un'infrastruttura IT e i suoi utenti, poiché possono essere confinate garantendo la flessibilità necessaria tipica di un PC basato sull'ormai consolidata architettura x86. Tra le public cloud più di successo sicuramente si può menzionare quello di Amazon [2] che offre macchine virtuali piuttosto che una piattaforma per lo sviluppo di applicazioni, come invece accade per Google Apps Engine [3]. Anche Azure [4], la public Cloud di Microsoft, era partita con un modello analogo a quello del Cloud di Google ma ha poi introdotto il cosiddetto VMRole, che consente di allocare macchine virtuali ad uso generale.

Il progetto Octopus nasce nel 2010 con l'idea di realizzare un sistema capace di orchestrare una rete di *hypervisor* con un duplice obiettivo: consentire un modello self-service di provisioning di macchine virtuali attraverso il Web e orchestrare le macchine virtuali, cercando di ottimizzare il consumo energetico con lo spostamento delle macchine virtuali in modo da liberare nodi fisici che possano essere spenti. Il sistema si è poi evoluto per investigare come generalizzare il sistema di regole necessarie a gestire i vincoli nell'allocazione delle risorse e la loro gestione ottimale attraverso l'introduzione di un sistema esperto.

In questo articolo illustreremo l'architettura di Octopus originale e come l'introduzione del sistema esperto CLIPS [5] ne abbia condizionato la struttura e la scalabilità.

2. Octopus

Octopus ha l'architettura mostrata in Figura 1: il sistema si occupa di organizzare e amministrare macchine virtuali in esecuzione su uno o più hypervisor Hyper-V di Microsoft; gli utenti possono creare VM ed amministrare quelle già create attraverso un'interfaccia Web (vedi Figura 2). L'accesso a una macchina virtuale avviene attraverso il portale Web che consente di collegarsi a macchine Unix mediante SSH e a macchine Windows utilizzando il protocollo RDP.

Il sistema è programmato in F# e controlla gli

Workshop GARR Calcolo e Storage Distribuito - Selected Papers. Roma 29-30 novembre 2012
ISBN: 978-88-905077-4-8 © 2013 Associazione Consortium GARR

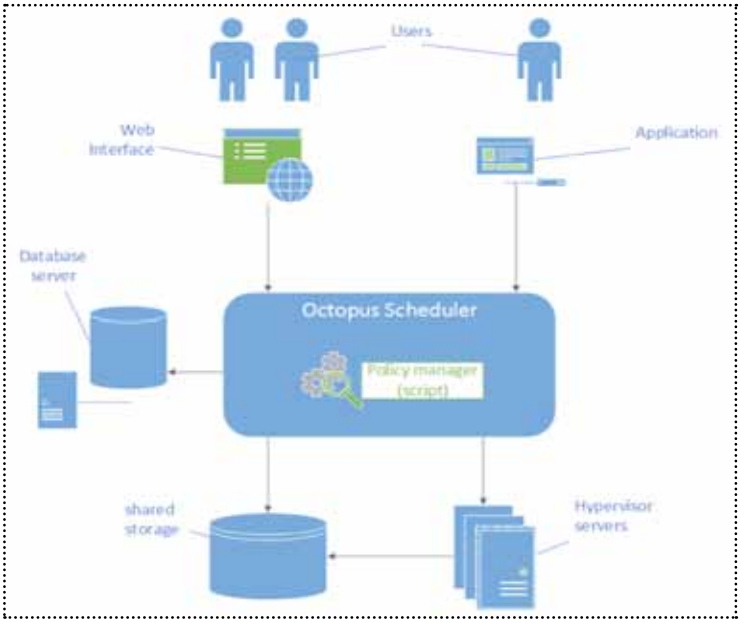


Fig 1 - Architettura di Octopus

Hyper-V utilizzando l'interfaccia WMI. La scelta di utilizzare lo stack Microsoft per la virtualizzazione è dovuta alla semplicità con cui Hyper-V si può controllare da programma e ha permesso di concentrare gli sforzi sulla realizzazione del sistema. Il linguaggio F# è Open Source e può essere eseguito su qualunque piattaforma su cui sia disponibile mono o .NET: è quindi possibile generalizzare il sistema ad altri hypervisor, mediante la sostituzione delle chiamate WMI con script che offrano funzionalità equivalenti.

Il cuore del sistema è il policy manager, ovvero un modulo responsabile di decidere dove allocare una nuova macchina virtuale, la quantità di risorse che possono essere dedicate ad un particolare utente, e in quali condizioni sospendere oppure spostare macchine virtuali da un hypervisor ad un altro della rete. Come già detto questo elemento era stato concepito con particolare attenzione agli aspetti di *green computing*, cercando quindi di distribuire il carico in modo da poter mettere in *standby* i nodi non necessari in un certo momento.

Il gestore del sistema Octopus esercita quindi il proprio controllo sulle politiche per l'allocazione delle risorse attraverso la definizione di due meccanismi essenziali:

- Un insieme di regole
- Un *repository* di immagini delle VM che gli utenti possono usare per creare nuove macchine virtuali.

Un'altra funzione esercitata da Octopus è l'organizzazione dei dischi delle VM utilizzando la tecnica dei *differencing disks*, cioè la creazione di nuovi dischi virtuali da associare a una VM utilizzando un'immagine già pronta come base. Il database di Octopus mantiene tutte le informazioni necessarie alla configurazione e al processo di orchestrazione effettuato dal gestore delle politiche del sistema.

3. Orchestrazione mediante un sistema esperto

La sperimentazione con una prima versione del sistema Octopus ha fatto emergere quali fossero le esigenze di un sistema di amministrazione di Cloud di virtual machine. Presto ci si è resi conto che per poter esprimere regole complesse occorreva riprogettare il modello per la definizione del gesto-

Fig 2 - L'interfaccia Web con le macchine virtuali create da un utente

re di politiche.

L'esplosione del numero di elementi da gestire in un'infrastruttura IT, anche a causa della virtualizzazione delle risorse, rende necessario l'impiego di automatismi sempre più sofisticati, ed è necessario riuscire a catturare la conoscenza di un IT manager per incorporarla in un sistema automatico di cui si possa controllare il funzionamento. Per questo motivo abbiamo deciso di integrare un vero e proprio sistema esperto in Octopus piuttosto che continuare lo sviluppo di un sistema a regole ad hoc la cui struttura era destinata a complicarsi, sfruttando soluzioni sviluppate nel campo dei sistemi esperti. Anche sistemi che gestiscono hypervisor, come ad esempio System Center 2012 di Microsoft, includono soluzioni basate su orchestrazione di processo, simili a quelle dei sistemi esperti che però non definiscono il comportamento in caso di conflitto tra più regole. La possibilità di gestire conflitti con opportune politiche, ad esempio decidendo, quali regole abbiano priorità, consente di esprimere in modo semplice politiche articolate.

Al crescere dei moduli di un'infrastruttura IT, e di conseguenza delle regole per la loro gestione, diviene sempre più difficile assicurarsi che un insieme di regole sia consistente; inoltre questi moduli risultano spesso difficili da estendere, perché sviluppati come interpreti che crescono progressivamente per estendere l'insieme di funzionalità supportate.

I sistemi esperti consentono agli specialisti del dominio di codificare la propria conoscenza specifica, che poi un sistema automatico provvederà ad impiegare per operare. I sistemi esperti sono stati studiati nell'ambito dei sistemi intelligenti ed è presente molta letteratura sul tema, prevalentemente sviluppata durante gli anni '80.

4. CLIPS

Un famoso sistema esperto, sviluppato alla NASA ed utilizzato in molti progetti, è CLIPS, un sistema a regole basato su un'implementazione efficiente dell'algoritmo RETE, la cui sintassi è un dialetto del linguaggio di programmazione LISP.

CLIPS, scritto in C, è progettato per essere incluso in applicazioni, ed esistono numerosi wrapper che consentono di includerlo in ambienti di programmazione moderni come ad esempio .NET. Il sistema consente di definire regole che vengono attivate da fatti espressi con asserzioni che rappresentano una data realtà. Ad esempio si può asserire che una certa VM è in stato di idle:

```
(assert (octopus-vm-idle "MyVM"))
```

La seguente regola definisce che quando una macchina virtuale sia in stato idle (qualunque cosa questo significhi) va sospesa la sua esecuzione:

```
(defrule suspend-when-idle (octopus-vm-idle ?x) =>
    (if (octopus-suspend-vm ?x) then
        (printout t ?x " suspended") else
        (printout t ?x " failed to suspend")))
```

Con l'introduzione di CLIPS in Octopus è il motore del sistema ad asserire automaticamente lo stato sotto forma di fatti in CLIPS. Le regole, predefinite o definite dall'amministratore del sistema, sono quindi attivate automaticamente da CLIPS e possono invocare azioni che Octopus espone come funzioni CLIPS. L'ambiente consente anche l'interazione con CLIPS attraverso un editor guidato dalla sintassi, che mostra le funzioni disponibili con l'ormai familiare approccio del completamento delle possibili voci mentre si scrive.

Mediante regole CLIPS, Octopus controlla i seguenti aspetti:

- Politiche di allocazione delle risorse agli utenti
- Monitoring dello stato delle macchine gestite
- Spostamento delle VM e spegnimento nodi per ottimizzare l'assorbimento energetico

5. Usare il sistema

Una volta installato Octopus su un nodo di un dominio Windows, è necessario allocare uno o più nodi dotati di hyper-V (disponibile anche nella versione desktop del sistema operativo a

partire da Windows 8). Vanno poi preparate le immagini di dischi virtuali, che saranno utilizzati come base per istanziare le macchine virtuali: il sistema supporta sia Windows che Linux.

Una volta configurato, il sistema accetta richieste attraverso il Web applicando le politiche definite nel file di configurazione del sistema e che contiene le regole scritte nel linguaggio CLIPS. Il manager IT può osservare il funzionamento interagendo in modo interattivo con la console CLIPS; può inoltre aggiungere regole anche temporaneamente per modificare il comportamento del sistema.

6. Conclusioni e sviluppi futuri

Octopus è un progetto open source (http://octopus.codeplex.com) ed è stato usato come campo di prova per studiare l'organizzazione di un meta hypervisor che offra meccanismi di self-service provisioning via Web vincolato da opportune politiche di gestione delle risorse. L'utilizzo di un sistema esperto ha consentito i realizzare un'architettura software del sistema più organica che permette di modellare esplicitamente la conoscenza del dominio di un esperto IT manager.

Il limite principale del sistema, oltre al fatto di essere ancora in uno stato prototipale, è quello di funzionare solo sulla piattaforma Microsoft. Stiamo lavorando per rendere gli hypervisor gestiti un parametro del sistema; l'adozione del sistema esperto in questo caso può favorire la gestione di ambienti eterogenei, in cui alcune operazioni possono essere disponibili solo tra hypervisor omologhi.

Riferimenti bibliografici

[1] Armbrust M., Fox A., Griffith R., Joseph A.D., Katz R., Konwinski A., Lee G., Patterson D., Rabkin A., Stoica I., and Zaharia M.; A view of Cloud Computing, Communications of the ACM, April 2010, 53:04.

[2] Amazon AWS, http://aws.amazon.com

[3] Google App Engine, http://appengine.google.com

[4] Microsoft Azure, http://windowsazure.com

[5] Giarratano J.C., Riley G.; Expert Systems, PWS Publishing Co., Boston, 1998, ISBN: 0534950531

Antonio Cisternino
cisterni@di.unipi.it
E' ricercatore presso il Dipartimento di Informatica dell'Università di Pisa e membro del centro interdipartimentale IT Center. Si occupa di programmazione di sistemi complessi, ambienti di virtualizzazione e user-interaction. È attivo nella comunità .NET a partire dal 2001 e più recentemente è uno dei leader della comunità del linguaggio di programmazione F#.

Maurizio Davini
mau@df.unipi.it
E' il Coordinatore Tecnico del Centro Interdipartimentale IT Center dell'Università di Pisa. Fino al 2012 è stato il responsabile del Centro di Calcolo del Dipartimento di Fisica. Ha fatto parte di numerosi advisory board in ternazionali in ambito HPC, collaborando con aziende quali Ferrari, AMD, Intel, Dell, e HP.

Marco Mura
mura@di.unipi.it
Ha una Laurea Specialistica in Tecnologie informatiche, Octopus è stato l'oggetto del lavoro di tesi. Ha partecipato alle attività dell'IT Center, partecipando con Acer a International Super Computing 2010. Nel 2010 ha svolto una internship presso Microsoft Research (Redmond) lavorando su tematiche relative ai fabric.