DHT Symphony Simulation

Andrea Aquino, Michele Consiglio Università di Bologna

10 gennaio 2013

Il protocollo DHT Symphony discusso da Manku, Bawa e Raghavan si propone come nuova frontiera per il mantenimento e l'accesso di dizionari distribuiti sulla rete globale. Sfruttando una topologia di rete virtuale ad anello sulla quale vengono instaurate connessioni tra nodi distanti, è possibile ridurre drasticamente i tempi di recupero di dati evitandone l'eccessiva replicazione. Tuttavia il numero di messaggi necessari al funzionamento del protocollo stesso non è oggetto di studio del documento prodotto dagli autori di cui sopra. Si intende quindi analizzare tale aspetto mediante la simulazione del protocollo a mezzo del simulatore Omnet++. Verrà inoltre introdotta una modifica protocollare con lo scopo di diminuire il numero di messaggi transitanti nella rete senza inficiare la correttezza e l'efficacia del protocollo.

Panoramica sulle tecnologie utilizzate

Al fine di completare l'installazione del software Open Nebula sono stati impiegati due calcolatori portatili le cui caratteristiche sono brevemente riassunte di seguito:

- OS: Ubuntu 11.10 (Oneiric Ocelot), 32 bit.

- CPU:
  - frontend: Intel® Core™2 Duo CPU T6670 @ 2.20GHz × 2
  - client: Intel® Core i5-460M da 2,53 GHz.
- Ram: 4 GB DDR3
- Virtualizzazione: Supporto nativo alla virtualizzazione hardware (vmx).

Se non diversamente specificato le caratteristiche di cui sopra sono da riferirsi ad entrambi i calcolatori.

Il software di rilievo

Il programma Open Nebula utilizza una vasta gamma di programmi e servizi open source disponibili per i sistemi Linux. Tra questi spiccano qemu, libvirt, ssh, apparmor. Una serie di test ha evidenziato la necessità di utilizzare le versioni più recenti di tali programmi, con particolare riguardo per qemu (v. 0.15.1). A tale scopo è stata effettuata la compilazione ed installazione della suddetta versione di qemu mediante gcc sia sulla macchina client che sulla macchina frontend mediante un procedimento della durata di circa 45 minuti.

Installazione e configurazione

L'installazione del software Open Nebula (v. 3.2.1) è stata effettuata estraendo l'archivio tar prelevato dal sito web *http://opennebula.org/*, configurandolo mediante ruby scons e compilandolo opportunamente. Per poter compilare correttamente è stato necessario installare una serie di librerie aggiuntive: g++, libxmlrpc-c3-dev, scons, libsqlite3-dev, libmysqlclient-dev, libxml2-dev, libssl-dev, ruby. I comandi seguenti riassumono i passi necessari per l'installazione del programma sulla macchina client e sulla macchina frontend:

- sudo apt-get install g++ libxmlrpc-c3-dev scons libsqlite3-dev libmysqlclient-dev libxml2-dev libssl-dev ruby
- scons -j2
- Frontend) sudo ./install.sh -d /srv/cloud/one -u oneadmin -g cloud
- Client) sudo ./install.sh -d /srv/cloud/one -u oneadmin -g cloud -c
- /srv/cloud/one/share/install_gems sunstone cloud

Sulle macchine è stato poi configurato l'account di amministrazione di Open Nebula (oneadmin) ed il gruppo relativo (cloud), ed è stata presa nota dell'uid e dell'gid. In particolare per semplicità all'utente oneadmin è stata associata la password test.

- groupadd cloud
- useradd -d /srv/cloud/one -g cloud -m oneadmin
- sudo passwd oneadmin

In seguito sono state esportate le variabili globali di riferimento del softare Open Nebula. Per comodità le istruzioni relative sono state introdotte nel file di configurazione del terminale bash (.bashrc) in modo da effettuarne l'esecuzione su ogni terminale attivo. Inoltre è stato ribadito che l'utente oneadmin ha password test per mezzo del file di configurazione one_auth.

- echo
  "export PATH=$PATH:/var/lib/gems/1.8/bin:
  /srv/cloud/one/bin
  export ONE_LOCATION=/srv/cloud/one
  export ONE_XMLRPC=http://localhost:2633/RPC2
  export ONE_AUTH=/srv/cloud/one/.one/one_auth"
  >> /.bashrc
- echo "oneadmin:test" > /src/cloud/one/.one/one_auth

Poichè il sottoalbero del filesystem /srv/cloud/one/ è stato destinato ad essere la home dell'utente oneadmin i file e le directory in esso contenute sono state assegnate a tale utente: sudo chown -R oneadmin:cloud /srv/cloud/one/

Infine, sebbene non fosse strettamente necessario, abbiamo garantito permessi di root all'utente oneadmin. Tale scelta semplifica lo sforzo di configurazione permettendo di effettuare modifiche ai file di configurazione protetti anche se loggati con tale utente. Pertanto abbiamo editato il file di dichiarazione dei sudoers mediante il comando visudo introducendo la linea: oneadmin ALL=(ALL:ALL) ALL

Dopo aver acceduto al sistema come utente oneadmin a mezzo del comando: su oneadmin, è stato riscontrato un errore di configurazione della shell di base e della home directory di tale utente. Di conseguenza mediante il comando chsh è stata assegnata una shell bash a oneadmin permettendo il funzionamento del login. A tal punto sono stati avviati i demoni di gestione dell'ambiente Open Nebula:

- one start

- oneacctd start
- sunstone-server start

Il servizio web sunstone è quindi stato reso disponibile mediante browser (ampiamente testato su Chromium Browser) all'url: http://127.0.0.1:9869

La configurazione degli host per Open Nebula è stata effettuata a mezzo terminale per permettere la configurazione dei parametri relativi ai driver delle macchine virtuali in uso, del filesystem distribuito (con particolare riguardo per nfs) e della rete. Per far ciò è stata dichiarata una entry nel file /etc/hosts che associa al nome andryak l'indirizzo ip 10.42.43.1 e al nome host1 l'indirizzo ip 10.42.43.12. Dopo tale specifica il comando: onehost create host1 im_kvm vmm_kvm tm_shared dummy, permette la creazione di un host virtuale relativo alla macchina fisica in ascolto sull'indirizzo specificato per host1. La medesima procedura è stata effettuata per configurare l'host andryak in quanto, per scarsità di risorse, abbiamo dovuto incorporare nel frontend anche le caratteristiche di un client Open Nebula.

Dopo aver realizzato un'immagine disco di una macchina virtuale Ubuntu 10.04 Lucid Lynx (tty only) mediante la procedura descritta nel seguente paragrafo vi sono stati applicati i massimi permessi (777), l'utente possessore oneadmin ed il gruppo di appartenenza cloud. Mediante il servizio web sunstone tale immagine disco è stata caricata e configurata come immagine raw per Open Nebula, le sono stati assegnati 512 MB di memoria RAM ed un massimo consumo del 50% di CPU. Infine è stato configurato un canale vnc verso l'indirizzo di loopback 127.0.0.1 alla porta 5930 per poter monitorare lo stato della macchina virtuale in esecuzione.

In seguito dopo che la macchina client ha effettuato il mounting del sottoalbero in cui opera Open Nebula (/srv/cloud/one) mediante il file system distribuito nfs è stato possibile effettuare il deploy della macchina e la sua migrazione (e migrazione live) dalla macchina host alla macchina frontend e viceversa. La procedura di configurazione del filesystem distribuito è argomento del paragrafo Configurazione nfs.

Disco virtuale per Ubuntu 10.04 tty

Per poter procedere alla creazione del disco virtuale contenente l'installazione di Ubuntu 10.04 tty è stato necessario innanzi tutto installare Qemu (versione 1.0.1). Una volta scaricato il pacchetto di installazione dall'indirizzo http://wiki.qemu.org/Download, è stato decompresso e sono state eseguite da terminale le seguenti istruzioni:

- sudo apt-get remove qemu qemu-kvm
- sudo apt-get install libglib2.0-dev zlib1g-dev
- ./configure –prefix=/usr
- make
- sudo make install

È stato quindi scaricato il pacchetto di installazione di Qemu-kvm 1.0 dall'indirizzo http://sourceforge.net/projects/kvm/files/qemu-kvm/1.0/qemu-kvm-1.0.tar.gz/download,estratto in una cartella ed installato a mezzo terminale tramite i seguenti comandi:

- ./configure –prefix=/usr
- make
- sudo make install

A questo punto è stato creato un disco virtuale vuoto tramite il comando

- qemu-img create -f raw ubuntu.img 3G

dove 3G sta per 3 Gigabyte.
Successivamente è stata avviata la macchina virtuale con il suddetto disco virtuale per installare il sistema operativo con il comando

- qemu -m 512 -hda ubuntu.img -cdrom ubuntu10-04.iso -boot d

dove "-cdrom" è la flag per lanciare il sistema operativo da iso,"-hda" dichiara l'hard disk virtuale di riferimento e "-m" serve ad indicare il quantitativo di ram da destinare alla macchina virtuale. Il suddetto comando ha però generato un errore (pci_add_option_rom: failed to find romfile "pxe-rtl8139.bin") che è stato risolto installando il pacchetto kvm-pxe.
Infine è stata avviata la macchina virtuale per testare il corretto funzionamento della stessa successivamente all'installazione del sistema operativo tramite il comando

- qemu-system-i386 -m 1024M -hda ubuntu.img

Configurazione nfs

Per poter rendere funzionante il file system distribuito nfs sono stati installati il server ed il client nfs rispettivamente sull'host che funge da front-end e su quello che funge da nodo del cluster.
Quindi è stato modificato il file /etc/exports nella macchina che funge da server, al cui interno è stata aggiunta la seguente riga:

- /srv/cloud/one *(rw,async,
  no_root_squash,
  no_subtree_check,anonuid=1001,anongid=1001)

che, in pratica, specifica di rendere disponibile la directory /srv/cloud/one con tutte le sottodirectory in modalità rw,che l'utente root nella macchina client accederà ai file come utente root anche sul server,con la verifica della presenza dei file richiesti nella corretta porzione del volume disabilitata e con userid e groupid corrispondenti a quelli di oneadmin e cloud in entrambe le macchine.
Fatto ciò si esporta la directory desiderata tramite il comando

- sudo exportfs -a

Il client, invece, dovrà semplicemente montare la directory desiderata prima di effettuare qualunque altra operazione, tramite il comando

- mount -t nfs andryak:/srv/cloud/one /srv/cloud/one

che indica di montare,tramite nfs, la directory esportata dalla macchina andryak all'interno della rispettiva directory nel client.

JBoss Application Server

All'interno della macchina virtuale la cui creazione è stata discussa nel paragrafo "Disco virtuale per Ubuntu 10.04 tty" è stato installato l'application server JBoss nella sua versione più recente (v. 7.1). Dopo aver scaricato l'archivio relativo ed averlo decompresso è stato avviato lo script di esecuzione (standalone) dell'application server configurato sulla porta 8080 all'indirizzo di loopback 127.0.0.1. E' stata creata una semplice applicazione, prendendo spunto da quella fornita da JBoss, in modo da far apparire il classico messaggio di test "Hello World". Infine attraverso il browser testuale elinks è stato possibile accedere al servizio web e dimostrare l'effettiva riuscita della configurazione di sistema. E' stata anche verificata l'effettiva persistenza del servizio in seguito a numerose migrazioni (live) della macchina virtuale per mezzo del software Open Nebula da un host ad un altro.

Bibliografia

- http://opennebula.org/documentation:archives:rel3.2, [ultima visita] Lun. 14 Mag. 2012
- http://www.scons.org/wiki/
- https://docs.jboss.org/author/display/AS71/Documentation
- http://wiki.qemu.org/Manual
- http://www.ubuntu.com/download/desktop/alternative-downloads
- https://help.ubuntu.com/community/KVM