How build a Fedora HPC cluster running OpenFoam in parallel, using Torque/PBS, OpenMPI, Host-based authentication and NFS

Pier Paolo Ciarravano

19/07/2010

Descrizione dell'architettura

Il cluster si compone di una macchina denominata MASTER NODE e di più macchine denominate SLAVE NODES (compute nodes). Il master node può a sua volte essere uno slaves node ed essere una risorsa per l'esecuzione dei job; nel caso del cluster che andremo a installare e configurare il master node sarà di questo tipo.

Tutte le macchine installate e configurate utilizzano Linux Fedora 12 (o versioni successive).

Il cluster utilizzerà Torque per la gestione dei job ed il master node sarà la macchina che si occuperà di eseguire il server e lo scheduler Torque.

Su tutti i nodi sarà installato OpenMPI con supporto Torque, per permettere ai comandi MPI di eseguire i processi andando ad identificare automaticamente i cores disponibili.

Inoltre per permettere l'esecuzione dei job tra i vari nodi, si configurerà una Host-based authentication in SSH.

Si è scelto di utilizzare NFS come file system condiviso, per non installare soluzioni commerciali o difficilmente configurabili, ma si può scegliere qualsiasi altra soluzione maggiormente performante e/o disponibile.

Installazione di Fedora 12 e configurazione

Procedere ad una normale installazione di Fedora 12 su tutti i nodi, includendo i pacchetti di sviluppo e escludendo i pacchetti server e office productivity.

Assegnare al masternode il nome:
`testmaster`

e ai nodi slave i nomi:
`testnode1`
`testnode2`
`testnode3`
…….

Modificare, per ogni nodo, il file `/etc/hosts` con la lista di tutti gli IP dei nodi e i nomi assegnati ai nodi:

```
X.X.X.X testmaster
X.X.X.X testnode1
X.X.X.X testnode2
……..
```

Modificare, per ogni nodo, il file `/etc/sysconfig/network` con la seguenta riga:

```
hostname=nome_nodo
```

Per eliminare il caricamento della modalità grafica X11 modificare, per ogni nodo, il file `/etc/inittab` con la seguenta riga finale:

```
id:3:initdefault:
```

Per consentire una maggior numero di colonne e righe nel terminale testuale e visualizzare tutto il loog di boot, per ogni nodo, modificare il file `/boot/grub.conf` nel seguente modo:

~~`rhgb quiet`~~ `vga=773`

eliminando il testo barrato.

Disattivare, per ogni nodo, SELinux modificando il file /etc/selinux/config con il seguente parametro:

```
SELINUX=disabled
```

e riavviate tutti i nodi. Per verificare che effettivamente SELinux sia correttamente disattivato, scrivere uno script eseguibile con le seguenti righe:

```
#!/bin/bash
selinuxenabled
echo $?
```

ed eseguirlo: se ritorna `1` allora SELinux è correttamente disattivato, se ritorna `0` allora SELinux è ancora attivo.

Se l'interfaccia di rete dei nodi non è attiva, eseguire il comando:

```
ifup eth0
```

Impostare il firewall del master node modificando il file `/etc/sysconfig/iptables` nel seguente modo:

```
# Firewall configuration written by system-config-firewall
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 15000:15004 -j ACCEPT
-A INPUT -p udp -m udp --dport 15000:15004 -j ACCEPT
-A INPUT -s testnode1 -p tcp -m tcp --dport 1024:65535 -j ACCEPT
-A INPUT -s testnode2 -p tcp -m tcp --dport 1024:65535 -j ACCEPT
-A INPUT -s ........ -p tcp -m tcp --dport 1024:65535 -j ACCEPT
........
-A INPUT -p tcp -m tcp --dport 2049 -j ACCEPT
-A INPUT -p udp -m udp --dport 2049 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 111 -j ACCEPT
-A INPUT -p udp -m udp --dport 111 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

Impostare il firewall dei nodi slaves modificando il file `/etc/sysconfig/iptables` nel seguente modo:

```
# Firewall configuration written by system-config-firewall
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 15000:15004 -j ACCEPT
-A INPUT -p udp -m udp --dport 15000:15004 -j ACCEPT
-A INPUT -s testmaster -p tcp -m tcp --dport 1024:65535 -j ACCEPT
-A INPUT -s testnode1 -p tcp -m tcp --dport 1024:65535 -j ACCEPT
-A INPUT -s testnode2 -p tcp -m tcp --dport 1024:65535 -j ACCEPT
-A INPUT -s ........ -p tcp -m tcp --dport 1024:65535 -j ACCEPT
........
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

Riavviare il servizio iptables su tutti i nodi (master e slaves) con il comando:

```
/etc/init.d/iptables restart
```

Configurazione utente e Host-based authentication

Creare, su ogni nodo, un utente con il nome `testuser` utilizzando il comando adduser e specificando il parametro per la creazione di una cartella utente in `/home`

Modificare, su ogni nodo, il file `/etc/ssh/ssh_config` con le seguenti proprietà:

```
HostbasedAuthentication  yes
EnableSSHKeysign         yes
```

Modificare, su ogni nodo, il file `/etc/ssh/shosts.equiv` aggiungendo i nomi di tutti i nodi del cluster:

```
testmaster
testnode1
testnode2
testnode3
.........
```

Modificare, su ogni nodo, il file `/etc/ssh/sshd_config` con le seguenti proprietà:

```
IgnoreRHosts                 no
IgnoreUserKnownHosts         yes
HostbasedAuthentication      yes
```

Su ogni nodo, posizionarsi nella directory `/etc/ssh`, ed eseguire come `root` i comandi di seguito, per importare le chiavi pubbliche degli altri nodi:

```
ssh-keyscan  -t  dsa  testmaster  >> ssh_known_hosts
ssh-keyscan  -t  dsa  testnode1   >> ssh_known_hosts
ssh-keyscan  -t  dsa  testnode2   >> ssh_known_hosts
ssh-keyscan  -t  dsa  testnode3   >> ssh_known_hosts
ssh-keyscan  -t  dsa  …….   >> ssh_known_hosts
```

Riavviare il servizio `sshd` con il comando:

```
/etc/init.d/sshd restart
```

Per testare la corretta configuarazione della Hostbased Authentication su tutti i nodi, loggarsi come utente `testuser` su un nodo (es:`testnode1`), ed eseguire i comandi

```
ssh –v testnode2
ssh –v testnode3
……
```

Se la configurazione è stata eseguita correttamente, il login ssh dovrebbe avvenire senza la richiesta di password.

Configurazione di NFS

Loggarsi sul nodo master (`testmaster`) come utente `testuser` e creare la cartella `/home/testuser/nfsrepo` che conterrà il file system fisico condiviso da NFS.

Loggarsi sul nodo master (`testmaster`) come utente `root` e modificare o creare il file `/etc/exports` con il seguente contenuto:

```
/home/testuser/nfsrepo testmaster(rw) testnode1(rw) testnode2(rw)
    testnode3(rw)……
```

Riavviare il servizio `nfs` con il comando:

```
/etc/init.d/nfs restart
```

Su tutti i nodi, compreso il nodo master, loggarsi come utente `testuser` e creare la cartella `/home/testuser/nfsmount`

Successivamente su tutti i nodi, compreso il nodo master, loggarsi come utente `root` ed eseguire il seguente comando per montare il file system NFS condiviso:

```
mount testmaster:/home/testuser/nfsrepo /home/testuser/nfsmount
```

Per evitare di dover ogni volta eseguire il mount al riavvio dei nodi, si può aggiungere una riga nel file `/etc/fstab` che specifica il montaggio automatico del file system: vedere documentazione di fstab.

Installazione e configurazione di Torque/PBS

Loggarsi sul nodo master (`testmaster`) come utente `root` ed eseguire i seguenti comandi:

```
yum install torque torque-client torque-server torque-mom
    torque-scheduler libtorque libtorque-devel

/etc/init.d/pbs_server stop

/usr/sbin/pbs_server – t create

/usr/share/doc/torque-2.1.10/torque.setup root
```

Creare il file `/var/torque/mom_priv/config` con il seguente contenuto:

```
$pbsserver testmaster
```

Creare il file /var/torque/server_priv/nodes con il seguente contenuto:

```
testmaster np=4
testnode1      np=4
testnode2 np=4
testnode3 np=4
……..
```

Dove il numero 4 andrà sostituito con il numero dei core disponibili su quel nodo.

Creare il file /var/torque/server_name con il seguente contenuto:

```
testmaster
```

Riavviare tutti i servizi installati con i comandi seguenti:

```
/etc/init.d/pbs_server   restart
/etc/init.d/pbs_mom      restart
/etc/init.d/pbs_sched    restart
```

Loggarsi sui nodi slaves (testnode1….) come utente `root` ed eseguire i seguenti comandi:

```
yum install torque torque-client torque-mom libtorque libtorque-devel
```

Creare il file `/var/torque/mom_priv/config` con il seguente contenuto:

```
$pbsserver testmaster
```

Creare il file /var/torque/server_name con il seguente contenuto:

```
testmaster
```

Riavviare il servizio mom con il comando:

```
/etc/init.d/pbs_mom      restart
```

Compilazione e installazione di OpenMPI

Su tutti i nodi, compreso il nodo master, loggarsi come utente `root` ed eseguire i seguenti comandi, dopo aver trasferito il file `openmpi-1.4.1.tar.gz` nella cartella `/root`:

```
cp  /usr/include/torque/*  /usr/include/

mkdir /usr/lib/openmpi

tar –xzf openmpi-1.4.1.tar.gz

cd openmpi-1.4.1

./configure --with-tm=/usr --prefix=/usr/lib/openmpi

make

make install
```

Per testare la corretta compilazione e installazione di OpenMPI con il supporto Torque eseguire il comando:

```
/usr/lib/openmpi/bin/ompi_info | grep –i tm
```

l'output ritornato dovrà contenere le seguenti righe:

```
MCA ras: tm (MCA v2.0, API v2.0, Component v1.4.1)
MCA plm: tm (MCA v2.0, API v2.0, Component v1.4.1)
```

Test di esecuzione di job MPI

Su un qualsiasi nodo, loggarsi come utente `testuser` e scompattare il file test_torque.zip nella cartella `/home/testuser/nfsmount/test_torque/` e successivamente lanciare il comando seguente per eseguire lo script `test.sh` come job sul cluster:

```
qsub –l nodes=N:ppn=M ./test.sh
```

dove N è il numero dei nodi sui quali si vuole eseguire il job e M è il numero di cores impiegati per ogni nodo che esegue il job.

Durante l'esecuzione dei job è possibile visualizzare lo stato della coda dei job e lo stato del job appena lanciato, con i sequenti comandi:

```
qstat

qstat –an

qstat –q
```

Al termine dell'esecuzione del job saranno disponibili due file all'interno della cartella `/home/testuser/nfsmount/test_torque/` uno con il nome `test.sh.eXX` contenente lo

standard error e uno con il nome `test.sh.oXX` contenente lo standard output (dove XX è il numero del job assegnato dalla coda).

E' possibile rimuovere e arrestare un job sulla coda eseguendo il seguente comando:

```
qdel XX
```

dove XX è il numero del job assegnato dalla coda.

Per eseguire un test che utilizzi MPI eseguire il seguente comando:

```
qsub -l nodes=N:ppn=M ./testHello.sh
```

dove N è il numero dei nodi sui quali si vuole eseguire il job e M è il numero di cores impiegati per ogni nodo che esegue il job. Nel file `testHello.sh.oXX`, se l'esecuzione con supporto MPI ha avuto buon esito, si potranno leggere i nodi impiegati dal job, con i rispettivi cores utilizzati.

E' disponibile anche il file sorgente del test MPI (hello.c), che all'occorrenza può essere ricompilato utilizzando la seguente procedura:

```
/usr/lib/openmpi/bin/mpicc hello.c -o hello
```

Notare che il comando `/usr/lib/openmpi/bin/mpirun` si occupa di eseguire un programma MPI (vedi file `testHello.sh`); notoriamente questo comando prende come parametro la lista dei nodi su cui si vuole far eseguire il programma MPI e il numero dei cores per nodo, ma utilizzando il supporto Torque con il comando qsub, queste informazioni sono automaticamente passate dalla coda Torque che esegue lo script, specificando le variabili `nodes` e `ppn`.

Con il comando `pbsnodes` è invece possibile visualizzare una descrizione dei nodi disponibili nella coda.

Istallazione di OpenFOAM ed esecuzione di un caso di test.

Scompattare il file `openfoam.tgz` e il file `data.tgz` nella cartella `/home/testuser/nfsmount/` creando rispettivamente le cartelle `openfoam` e `data`:

```
tar -xzf openfoam.tgz
tar -xzf data.tgz
mkdir data_backup
```

Copiare il file `run_openfoam.sh` nella cartella `/home/testuser/nfsmount/`

La cartella `openfoam` contiene il programma OpenFOAM, mentre la cartella `data` contiene i file dei dati del caso che si vuole analizzare.

Tramite lo script `run_openfoam.sh` viene eseguito il caso contenuto nella cartella `data`; Lo script si occupa di decomporre la mesh per i cores (processors) specificati nel momento

dell'esecuzione (decomposePar), dell'esecuzione di icoFoam, e successivamente della ricostruzione dei dati di output (reconstructPar).
Per eseguire pertanto OpenFoam sulla coda in parallelo sui nodi disponibili, lanciare il comando:

```
qsub -l nodes=N:ppn=M ./run_openfoam.sh
```

dove N è il numero dei nodi sui quali si vuole eseguire il job e M è il numero di cores impiegati per ogni nodo che esegue il job. Al termine del job, nel file `run_openfoam.sh.oXX`, si potrà leggere l'output della procedura, e nel file `run_openfoam.sh.eXX` gli eventuali errori in output.

Per verificare lo stato del job utilizzare il comando `qstat`.

Una volta terminato il job, nella cartella `data` saranno presenti i file di output di OpenFOAM, e nella cartella `data_backup` il file tar gzip di backup dei dati appena calcolati.

Tutte le cartelle `/home/testuser/nfsmount/` su tutti i nodi mostreranno naturalmente lo stesso contenuto.