CREAZIONE CLUSTER OPENMOSIX

Lo scopo di questo scritto é la tracciatura passo-passo delle operazioni necessarie alla creazione di un cluster openMosix, da adesso in poi indicato con i termini 'cluster' o 'pool'; i computer costituenti il cluster saranno chiamati 'nodi'.

A CHI SI RIVOLGE QUESTO DOCUMENTO

Mi rivolgo principalmente a persone come me, e cioé persone che usano Debian/Woody, che sono interessate all'argomento 'cluster' , che hanno già letto qualcosa sull'argomento e che quindi non sono digiuni della materia, ma che hanno avuto problemi ad implementare il proprio cluster.
Tuttavia, **nel caso in cui il lettore non conosca la materia**, fornirò una veloce e breve panoramica sull'argomento nelle prossime righe.

In giro per la Rete si possono trovare HOW-TO più o meno chiari, in italiano o in inglese, ma sono sempre dedicati a RedHat o altre distribuzioni. Poiché utilizzo Debian, ho deciso di scrivere questo documento che spiega come installare un cluster per la debian Woody.

Non ho la pretesa di spiegare per filo e per segno la storia di Mosix prima e openMosix poi, per queste informazioni, mi limito a rimandare il lettore allo "openMosix-HOWTO" (www.openmosix.org) ed alla documentazione reperibile in rete.

COSA E' OPENMOSIX?

openMosix é la versione GPL di Mosix: col tempo si è evoluto diventando un progetto a sè stante, replicando, migliorando ed espandendo le caratteristiche di Mosix.
OpenMosix é una patch del kernel linux, che lo mette in grado di far migrare i processi da un nodo ad un altro, accellerandone completamento.

COSA E' UN CLUSTER?

La definizione più generale di cluster é un insieme componenti hardware e software dedicati all'ottenimento di:

- servizi
- prestazioni
- gradi di affidabilità

non ottenibili dalle singole unità dell'insieme.

Qui prenderemo in esame le PRESTAZIONI, verrà cioé creato un **CLUSTER DI CALCOLO**

Basandosi sul concetto che "l'unione fa la forza" due o più PC correttamente configurati eseguiranno una serie di compiti in un tempo minore rispetto al singolo PC.

VANTAGGI DATI DALL'USO DI OPENMOSIX

SIMMETRIA:

Ogni elemento del gruppo (ogni **"nodo"** del **"pool"**) ha grado "paritetico" agli altri: ogni nodo può ricevere e/o fornire potenza di calcolo dai/agli altri nodi: in altre parole, se il PC1 può lavorare per il PC2, così il PC2 può lavorare per PC1.

ECONOMICITA':

Non é necessario hardware dedicato: normali PC connessi tramite schede di rete possono essere trasformati in cluster.

SCALABILITA':

Per aumentare le prestazioni, basta aggiungere nodi (max 65536) al pool.

TRASPARENZA:

Una volta in eseguzione, sarà il sistema stesso a bilanciare il carico di lavoro ed a ripartire i processi sulle varie macchine.

ADATTABILITA':

Non é necessario utilizzare macchine tutte uguali, ma possono essere usate varie tipologie di CPU (per il momonto solo x86), per cui posso mettere nel pool portatili, desktop, macchine più o meno potenti, come Pentium 166 o Athlon 1800+, etc.

FLESSIBILITA':

Poiché la partecipazione di un nodo al cluster può essere decisa tramite un semplice comando, é possibile dedicare il proprio pc al cluster solo se lo si desidera.

CONSIDERAZIONI SU OPENMOSIX (sul clustering in generale e cenni di programmazione)

Compilazioni, rendering, modellazione 3D, conversioni audio/video, analisi/elaborazione di grosse masse di dati... Sono tutti compiti che richiedono un enorme numero di calcoli e che richiedono ore ed ore di elaborazione su una singola macchina, possono essere rapidamente eseguite in poche decine di minuti su di un cluster oM. Uno dei punti di forza di oM é che non richiede che i programmi vengano scritti (o riscritti) appositamente per sfruttare la capacità di migrazione: é infatti oM che si incarica di ripartire i processi tra i vari nodi, in modo totalmente trasparente all'utente.
Al momento attuale (Kernel 2.4.20/21) oM permette la migrazione tra i nodi unicamente per i programmi che non fanno uso di memoria condivisa (shared memory); questo limita il numero dei processi migrabili. Per fare un esempio, mozilla e openoffice non migrano, per cui se avete in casa 4 o 5 vecchi Pentium 1 a 133Mhz e sperate di metterli in cluster per migliorare le prestazioni di KDE+mozilla, etc, scordatevelo :-)
Inoltre, c'è da tenere in considerazione il fatto che oM lavora a livello di 'processo': ogni processo è considerato una entità 'atomica' cioè NON ulteriormente suddivisibile.
Facciamo un esempio per chiarire la situazione, (assumendo che i processi richiedano la stessa potenza di calcolo e che i nodi del cluster siano uguali tra loro).
Se ci sono 8 processi che richiedono 60 minuti l'uno per essere eseguiti, un cluster di 8 PC potra' eseguire gli 8 processi in parallelo (uno per nodo del cluster) e alla fine gli 8 processi saranno completati in 60 minuti o poco più. Però se ho un unico processo che necessita di 8 ore (480 minuti) per essere completato, in questo caso il cluster non porterà nessun vantaggio poichè il processo in questione è 'atomico' cioè oM non può spezzarlo in più sottoprocessi da migrare su altri nodi. L'unico beneficio che oM può fornire è quello di individuare quale è la macchina più potente del cluster (nel caso di nodi non identici) ed inviare su questa macchina il processo più pesante, togliendo al contempo altri eventuali processi, per dedicare all'unico processo pesante tutta la capacità elaborativa del nodo.

A volte, occorre una decisione 'umana' su COME far eseguire determinati compiti al cluster: meglio lanciare tanti piccoli processi in parallelo (che verranno migrati su altri nodi) piuttosto che un unico grande processo (che NON beneficerà della potenza del cluster).

PREREQUISITI
Parto dal presupposto che le macchine abbiano tutte un indirizzo IP statico, con indirizzi che vanno da 192.168.1.10 a 192.168.1.17 e che siano connesse in rete tramite un HUB da 10Mbit/s. Il sistema operativo é GNU/Linux Debian Woody.
I nodi devono avere tutti lo stesso kernel e relativa patch oM, altrimenti non funzioneranno.

Io utilizzo il programma `mc`
(Midnight Commander) come file manager e per editare/modificare i file di testo, per cui vedrete che uso il programma `mcedit` ... se siete abituati ad altri edicor, come vi o vim, usateli pure.

Prima di proseguire, é bene tenere a mente che l'utilizzo di questo cluster comporterà una diminuzione della sicurezza delle singole macchine, in quanto saranno vulnerabili ad azioni di craking sia dall'esterno che da una macchina verso l'altra. Il sistema nasce per essere stand-alone e dedicato allo studio ed ai test di oM. Pur essendo questo cluster paritetico (ogni macchina può far lavorare le altre), per comodità durante i test userò una macchina come master e le altre come slave in attesa di ordini dal master. Rammento che la cosa che ora interessa é solo la messa in funzione del cluster di calcolo: non saranno prese in considerazione altre problemematiche come la diminuita sicurezza, il fatto che non vada la scheda audio o la presa USB, etc.

Qui si avranno un massimo di 8 nodi e avranno i seguenti nomi ed indirizzi IP:

NODO	NOME	INDIRIZZO IP	FUNZIONE
1	master	192.168.1.10	Master
2	nodo2	192.168.1.11	nodo slave
3	nodo3	192.168.1.12	nodo slave
4	nodo4	192.168.1.13	nodo slave
5	nodo5	192.168.1.14	nodo slave
6	nodo6	192.168.1.15	nodo slave
7	nodo7	192.168.1.16	nodo slave
8	nodo8	192.168.1.17	nodo slave

La connessione sarà garantita da un HUB a 8 porte, con velocità di 10Mbit/s.
Prima di dimenticarsene, attaccate subito in pc che vorrete abilitare al cluster all'HUB.

SOFTWARE NECESSARIO
servono i sorgenti PULITI, cioé senza patch. I sorgenti del kernel delle varie distribuzioni (Debian, slackware, etc) **NON** vanno bene, occorre procurarsi i sorgenti originali. Il sito da cui scaricare é (`www.kernel.org`).
Il kernel che verrà usato é l'ultimo dichiarato stabile, cioé il 2.4.20. Il file si chiama `linux-2.4.20.tar.bz2`.
La patch per abilitare il kernel al clustering é `openMosix-2.4.20-1.gz` reperibile partendo dal sito (`www.openmosix.org`)
; sempre nello stesso sito, procuratevi le utility per amministrare il cluster: l'ultima versione del file si chiama `openMosixUserland-0.2.4.tgz`

E FINALMENTE SI COMINCIA

La compilazione del kernel e la sua abilitazione é, limitatamente alla parte oM, identica per tutte i nodi del pool. Se non altrimenti specificato, le prossime operazioni andranno compiute su tutte i nodi del pool.

Anche se il cluster verrà configurato per 8 nodi, qui ne verranno creati solo 3 nodi, identici tra loro: si tratta di 3 portatili. Il motivo per cui verranno usati solo 3 computer é semplice: non ne ho di più :-)

MODELLO : IBM ThinkPad TP600
CPU : Pentium2 266Mhz
RAM : 128M o superiore
DISCO : 1G o superiore
RETE : Scheda a 10 o 100Mbit/s

Non starò qui a riportare quali opzioni attivare nel kernel, né mi occuperò della risoluzione di eventuali problemi di networking. Dò per scontato che il lettore abbia già una certa esperienza nell'uso di Debian, della ricompilazione del kernel: comunque, indico qui di seguito i passi necessari a ricompilare il kernel e modificare il lilo.

Ora darò di seguito tutta la lista di comandi per creare il kernel

```
# mv /directory_di_partenza/linux-2.4.20.tar.bz2 /usr/src/
# mv /directory_di_partenza/openMosix-2.4.20-1.gz /usr/src/
# cd /usr/src
# tar jxvf linux-2.4.20.tar.bz2
# rm linux
# ln -s linux-2.4.20 linux
# cd linux
# zcat /usr/src/openMosix-2.4.20-1.gz | patch -p1
# make menuconfig
```

Ora si presenta la scelta dei parametri di openMosix, configurare le opzioni come qui indicato:

```
[*] openMosix process migration support
[ ] Support clusters with a complex network topology
[*] Stricter security on openMosix ports
(3) Level of process-identity disclosure (0-3)
[*] openMosix File-System
[ ] Poll/Select exceptions on pipes
[ ] Disable OOM Killer
```

Configurate il resto del kernel, salvate ed uscite. Ora occorre compilare il kernel

```
# make clean
# make dep
# make clean
# make bzImage
# make modules
# make modules_install
# cp /usr/src/linux/arch/i386/boot/bzImage /boot/linux-2.4.20-oM
```

Il kernel é compilato e messo nella directory /boot, ora occorre modificare il /etc/lilo.conf ed assicuratevi che contenga una riga con la parola

```
prompt
```

senza il # davanti. Inoltre il file deve contenere queste righe, per abilitare il kernel ricompilato:

```
image=/boot/linux-2.4.20-oM
label=linux-2.4.20-oM
read-only
optional
```

Adesso aggiornate il lilo dando il comando:

```
# lilo -v
```

INSTALLAZIONE SCRIPT
Debian ci viene in aiuto, poiché esiste gia un pacchetto contenente lo script dal nome 'openmosix' che verrà installato in /etc/init.d/ e i link per attivare/disattivare il nodo alla partenza/stop.
E' sufficiente installare col solito comando:

```
# apt-get install openmosix
```

RICOMPILAZIONE COMANDI
Esistono dei comandi senza i quali il cluster non può essere configurato, monitorato ed amministrato. Qui di seguito le operazioni per crearli dai sorgenti.

```
# mv /directory_di_partenza/openMosixUserland-0.2.4.tgz /usr/src/
```

```
# cd /usr/src
# tar zxvf openMosixUserland-0.2.4.tgz
# cp openMosixUserland-0.2.4
# mcedit configuration
```

Assicuratevi che il file contenga queste scritte:

```
OPENMOSIX=/usr/src/linux
MONNAME=mmon
CC=gcc
INSTALLBASEDIR=/
INSTALLEXTRADIR=/usr
INSTALLMANDIR=$(INSTALLEXTRADIR)/share/man
CFLAGS = -Wall -I/m/include -I./ -I/usr/include -I$(OPENMOSIX)/include \
-Wpointer-arith --pedantic -Wcast-qual -Wconversion \
-Wwrite-strings -Wredundant-decls -O2
INSTALL=/usr/bin/install
```

A questo punto, digitate:

```
# make all
```

...e i comandi verranno compilati.

Ci saranno molti 'warning' in fase di compilazione: comunque, dovrebbe procedere tutto bene ed alla fine dovrebbe comparire un simpatico messaggio cher augura buon lavoro.

Ecco alcuni dei comandi creati, posti nella directory /bin:

```
mmon    : monitorizza lo stato del cluster
mosctl  : tool di amministrazione del cluster
mosrun  : esegue un comando su un particolare nodo del cluster
```

INSTALLIAMO MPS (e mtop)

Installatelo tramite:

```
# apt-get install mps
```

Il comando 'mps' é una versione potenziata di 'ps'. Questo comando aggiunge una colonna chiamata 'NODE' alla lista delle altre informazioni fornite da ps.
La colonna NODE sta ad indicare quale nodo del cluster il processo sta girando.

NOTA:
con 8 nodi il NODE varierà da 1 a 8, però il nodo su cui è attivo mps o mtop, verrà sempre indicato come nodo numero zero.

CONFIGURAZIONE SCHEDA DI RETE
per quei pochi che ancora non lo sanno, la configurazione della scheda di rete avviene grazie al file

```
/etc/network/interfaces
```

Ora, questo file di solito configura le schede di rete e la scheda di loopback. di solito la scheda di rete e' una sola (eth0).
Per chiarezza, allego il contenuto di /etc/network/interfaces che e' presente sul mio nodo master:

```
#questa e' l'interfaccia di loopback
auto lo
iface lo inet loopback

#questa e' l'interfaccia di rete
auto eth0
```

```
iface eth0 inet static
    address 192.168.1.10
    netmask 255.255.255.0
    network 192.168.1.0
    broadcast 192.168.1.255
```

Ovviamente, l'indirizzo 192.168.1.10 E' VALIDO SOLO PER IL NODO MASTER, sugli altri nodi sara' rispettivamente 192.168.1.11, 192.168.1.12, etc.

PREPARIAMO I NODI

```
# mkdir /mfs
```

Questo comando crea una directory, all'interno della quale verranno automaticamente create dal driver oM delle sotto-directory /1, /2, /3 etc, tante quante i nodi presenti nel cluster.

```
# mcedit /etc/fstab
```

assicuratevi che questo file contenga questa riga, che abilita il file-system oM.

```
mfs     /mfs    mfs     odfsa=1         0       0
```

Ultimo passo:

```
# mcedit /etc/openmosix.map
```

assicuratevi che questo file contenga gli indirizzi dei vari nodi: per comodità lo allego integralmente.

```
1  192.168.1.10  1
2  192.168.1.11  1
3  192.168.1.12  1
4  192.168.1.13  1
5  192.168.1.14  1
6  192.168.1.15  1
7  192.168.1.16  1
8  192.168.1.17  1
```

Adesso fate uno shutdown e al reboot scegliete `linux-2.4.20-oM`

PARTENZA CLUSTER
A questo punto, se non é attiva, configurate la scheda di rete con:

```
# /etc/init.d/networking restart
```

Poi lanciate lo script di openMosix con:

```
# /etc/init.d/openmosix start
```

Se compare un messaggio come il seguente:

setpe: the supplied table is well-formatted,
but my IP address (127.0.0.1) is not there!

Significa che l'hostmane del pc non figura nel file /etc/hosts con lo stesso indirizzo IP presente nel file /etc/openmosix.map.
Se per esempio il PC si chiama master, nel file /etc/hosts sarà presente una riga cosi:

```
127.0.0.1 master localhost
```

Occorre modificare il file nel seguente modo:

```
198.168.1.10 master
127.0.0.1 localhost
```

Rilanciate lo script di openMosix con:
```
# /etc/init.d/openmosix start
```

il sistema manda questi errori:

/etc/init.d/openmosix: /proc/mosix/admin/sspeed: No such file or directory
/etc/init.d/openmosix: /usr/bin/mosctl: No such file or directory
/etc/init.d/openmosix: /usr/bin/mosctl: No such file or directory
/etc/init.d/openmosix: /usr/bin/mosctl: No such file or directory
/etc/init.d/openmosix: /usr/bin/mosctl: No such file or directory

il primo errore dipende dal fatto che il file /proc/mosix/admin/sspeed non esiste ma che lo script deve invece usare /proc/hpc/admin/sspeed.
Il problema si risolve editando lo script /etc/init.d/openmosix e sostituendo le stringhe /proc/mosix con /proc/hpc.
Gli altri errori dipendono dal fatto che il file /usr/bin/mosctl non esiste in quella posizione ma in questa: /bin/mosctl. Il problema si risolve semplicemente creando un link simbolico

```
# ln -s /bin/mosctl /usr/bin/mosctl
```

Bene! Eseguite le modifiche del caso, spegnete e riaccendete i pc, e alla partenza, scegliete il kernel con oM (potete renderlo di default modificando oppurtunatamente il file /etc/lilo.conf). Per verificare il funzionamento del nodo, (sto lavorando sul master) digitate in comando:
```
# /etc/init.d/openmosix status
```

La risposta che ottengo é la seguente: (sto lavorando sul master)

```
This is OpenMosix node #1
Network protocol: 2 (AF_INET)
OpenMosix range     1-1     begins at master
OpenMosix range     2-2     begins at 192.168.1.11
OpenMosix range     3-3     begins at 192.168.1.12
OpenMosix range     4-4     begins at 192.168.1.13
OpenMosix range     5-5     begins at 192.168.1.14
OpenMosix range     6-6     begins at 192.168.1.15
OpenMosix range     7-7     begins at 192.168.1.16
OpenMosix range     8-8     begins at 192.168.1.17
Total configured: 8
```

Notate il fatto che oM viene attivato PRIMA che ci si logghi al sistema, quindi la potenza di calcolo dei nodi nodi sarà disponibile appena accesi, poiché oM funziona come un qualsiasi processo che gira in background.

ATTENZIONE!!!
Nel caso vi apparisse un messaggio di errore tipo: "this is not MOSIX!!", non dovrete fare altro che ricompilare i comandi, vedi piu' sopra il punto **RICOMPILAZIONE COMANDI**

ACCESSO AI FILESYSTEM DEI NODI
Esiste la possibilità di accedere al file-system presente sui nodi.
Se per esempio voglio rendere accessibile a tutti i nodi del pool il file-system del nodo X, allora dovrò editare, sul nodo X, il file /etc/default/openmosix ed abilitare la variabile in questo modo:

```
MFS=yes
```

Alla successiva partenza, il nodo X esporterà verso tutti gli altri nodi il proprio file-system. Gli altri nodi del pool potranno accedere al file-system del nodo X attraverso la directory /mfs/X. Questa possibilità é comoda, nel caso

si voglia condividere dei file (sorgenti kernel o altro), oppure amministrare e configurare i nodi da qualunque altro nodo ma é potenzialmente pericolosa in quanto c'é la possibilità di cancellare i nodi. Di default, l'opzione é disabilitata, al lettore la scelta di abilitarla o meno in funzione delle proprie necessità.

MONITORIAMO IL CLUSTER
Ci sono vari comandi (quelli che abbiamo compilato prima) che aiutano a monitorare, amministrare ed eseguire una fine taratura del sistema. Rimando al solito openmosix-HOWTO per una lettura approfondita.
Qui prenderemo in considerazione solo il comando

```
mmon
```

(nelle altre distribuzioni, questo comando viene chiamato 'mosmon') il quale permette di monitorizzare visivamente lo stato del cluster.
Dopo aver lanciato il comando, appare un grafico che indica il carico (load) dei nodi. I nodi attivi sono indicati dal numero ad essi corrispondente. E' possible avere informazioni premendo dei tasti:

```
q or Q Quit program
w/v/V/a horizontal (wide), vertical, super-vertical (tight) or
automatic selection of numbering
s show processor speeds, rather than loads
m show memory (used out of total) rather than loads
r show memory (raw used/free out of total) rather than loads
u show utilizability percentage rather than loads
l mostra il carico (default)
d mostra i nodi morti (configurati ma non operativi)
D l'opposto del tasto 'd' t mostra/toglie il numero dei nodi/CPU (opzione
lenta, sconsigliata su cluster grandi)
y mostra lo yardstick in uso (eg. velocita della CPU)
Enter ridisegna lo schermo
Se ci sono troppi nodi e non tutti stanno nella schermata:
freccia destra/sinistra muove di un nodo a destra/sinistra
n or p muove di una pagina a destra/sinistra
```

TESTIAMO IL CLUSTER
Occorre creare uno script di esempio, eccone uno che lancia 10 processi indipendenti (dei semplici cicli for-next). (Per chiarezza, rammento che sto sempre lavorando sul nodo master).

```
#!/bin/bash
#lancia 10 processi awk indipendenti
for n in 0 1 2 3 4 5 6 7 8 9 ; do
awk 'BEGIN { for (i=0;i<10000;i++) for (j=0;j<10000;j++);}' &
done
```

Salvate lo script, rendetelo eseguibile e lanciatelo.
Il programma mmon vi pervetterà di vedere i 10 processi 'sparpagliarsi' per i vari nodi del cluster.

Viene ora utile il comando:

```
mps a
```

che permette di vedere quanti processi girano e su quali nodi. Notate come nella colonna NODE i numeri varino da 0 a 3, indice del fatto che i processi sono ripartiti tra i 4 nodi del cluster.

```
PID TTY NODE STAT TIME COMMAND

281   2    0 S    0:00 /sbin/getty 38400 tty2
282   3    0 S    0:00 /sbin/getty 38400 tty3
283   4    0 S    0:00 /sbin/getty 38400 tty4
284   5    0 S    0:00 /sbin/getty 38400 tty5
```

```
286   6    0 S    0:00 /sbin/getty 38400 tty6
377  ?     0 S    0:00 /bin/cat
378   1    0 S    0:00 -bash
430  ?     0 S    0:00 /bin/bash
442  ?     0 S    0:00 /bin/bash
453  ?     0 R    0:00 awk BEGIN { for (i=0;i<10000;i++) for (j=0;j<10000;j++);}
454  ?     1 R    0:00 awk BEGIN { for (i=0;i<10000;i++) for (j=0;j<10000;j++);}
455  ?     2 R    0:00 awk BEGIN { for (i=0;i<10000;i++) for (j=0;j<10000;j++);}
456  ?     3 R    0:00 awk BEGIN { for (i=0;i<10000;i++) for (j=0;j<10000;j++);}
457  ?     3 R    0:00 awk BEGIN { for (i=0;i<10000;i++) for (j=0;j<10000;j++);}
458  ?     2 R    0:00 awk BEGIN { for (i=0;i<10000;i++) for (j=0;j<10000;j++);}
459  ?     1 R    0:00 awk BEGIN { for (i=0;i<10000;i++) for (j=0;j<10000;j++);}
460  ?     1 R    0:00 awk BEGIN { for (i=0;i<10000;i++) for (j=0;j<10000;j++);}
461  ?     0 R    0:00 awk BEGIN { for (i=0;i<10000;i++) for (j=0;j<10000;j++);}
462  ?     0 R    0:00 awk BEGIN { for (i=0;i<10000;i++) for (j=0;j<10000;j++);}
463  ?     1 R    0:00 mps a
```

Esiste anche 'mtop', la versione oM di 'top'.
Anche qui, l'unica differenza rispetto al comando originale, é l'agginta della colonna 'NODE'.
Ecco qui un esempio delle informazioni fornite da mtop:

```
PID USER     PRI  NI  SIZE  RSS SHARE STAT N# %CPU %MEM   TIME COMMAND
528 root      20   0   456  456   376 R     3 19.2  0.2   0:05 awk
530 root      20   0   456  456   376 R     2 18.2  0.2   0:05 awk
531 root      19   0   456  456   376 R     2 19.2  0.2   0:05 awk
532 root      19   0   456  456   376 R     3 19.1  0.2   0:05 awk
529 root      20   0   456  456   376 R     1 18.2  0.2   0:05 awk
508 root      14   0   904  904   716 R     0  1.1  0.4   0:03 mtop
429 root       9   0  9232 9232  7900 R     0  0.9  4.8   0:30 kdeinit
```

ESEMPIO TIPICO DI CREAZIONE CLUSTER

Supponiamo che abbiate a disposizione una dozzina di PC (aula didattica o ufficio di medie dimensioni) connessi in LAN tramite un HUB (meglio se uno switch): questi PC sono rapidamente (mezza giornata) convertibili in un cluster openMosix a costo zero! Ogni PC può avere il suo kernel con openMosix attivato, e l'utente del PC, tramite la manipolazione del file /etc/default/openmosix, è in grado di decidere se:
- abilitare o no il proprio PC come membro del cluster
- richiedere 'potenza elaborativa' al cluster
- fornire 'potenza elaborativa' al cluster
- rendere o no disponibile (esportare) il proprio file-system al cluster (molto comodo per condividere i sorgenti del kernel o altro, ma anche MOLTO pericoloso in quando da remoto possono agire sul filesystem del vostro PC, quindi: **MOLTA ATTENZIONE**)

Come precedentemente indicato, non tutti i programmi traggono beneficio dal clustering, poiché molti sono i fattori che entrano in gioco, come la tipologia di connessione e la velocità della rete, per esempio.
Nella mia particolare situazione, ho 3 portatili connessi tramite HUB a 10Mbit/s. Questa sistemazione é discreta per determinate situazioni, mentre potrebbe essere deleteria per altre. Supponiamo di dover svolgere questi due compiti:

- CASO_A - calcoli matriciali su 100 vettori (ogni vettore é grande 10KB e genera un risultato di 10KB)
- CASO_B - conversione di 100 file da WAV in MP3 (ogni file WAV é grande 50MB e genera un file MP3 di 5MB)

Per semplificare, supponiamo che:
- il tempo per calcolare un vettore sia uguale al tempo di conversione WAV->MP3 e che questo tempo sia di un minuto
- la banda passante tra un nodo e l'altro sia sempre e comunque di 512KB/s
Ora, se nel CASO_A i tempi di migrazione di un processo da un nodo all'altro é molto piccolo, nel CASO_B il tempo necessario al passaggio del file WAV in andata, sommato al tempo necessario del file MP3 di ritorno, sarebbe superiore al tempo necessario per la trasformazione! Avremmo in questo caso che il cluster ha lavorato più lentamente rispetto al singolo pc, se questo avesse lavorato da solo.
La soluzione più ovvia (ma costosa) é quella di procurarsi hardware più potente (schede di rete a 100Mb o

1000Mb e uno switch o un router per supportare tali velocità di comunicazione, hard disk veloci, molta ram, etc). Nel caso di elaborazione su grandi masse di dati (conversione di file sonori), si potrebbe prendere in considerazione la possibilità di creare un cluster di (per esempio) 5 nodi in cui ogni nodo sia connesso a tutti gli altri nodi: in questo caso la velocità di connessione tra i nodi sarebbe elevata, ma ci sarebbero due problemi:

1 - il cluster non potrebbe avere molti nodi (le moderne mainboard non hanno piu di 6 bus PCI)
2 - occorrerebbero molte schede di rete: con n nodi servirebbero n*(n-1) schede. 5 PC = 20 schede di rete.

In sintesi, il rapporto costi/benefici di un cluster dipende dal tipo di problemi che é chiamato a risolvere e da come viene fisicamente implementato.

Nel prossimo documento esamineremo le prestazioni di un cluster in funzione di vari parametri, quali gli applicativi usati, la velocita' e la topologia di rete (cioè il modo in cui i nodi sono connessi tra di loro). (NOTA: il documento e' pronto, ma non posso completarlo, poiche' non ho sufficente hardware (pc, etc) per poter eseguire dei tests sufficentemente accurati.... Qualcuno ha del vecchio hardware da darmi, tipo 8 Pentium a 200Mhz?)

CONCLUSIONI

Siamo arrivati alla fine di questo scritto e in cluster é attivo. Abbiamo visto come oM permetta di 'aggregare' un certo numero di computer per poter ottenere un cluster. Questa molteplicità di macchine viene vista come una unità singola dal punto di vista della potenza elaborativa. Come avevo detto all'inizio, alcuni punti dell'argomento oM sono stati solo accennati, ma sarebbe bene che venissero approfonditi attingendo ai rimandi a cui ho fatto riferimento all'interno di questo documento.

NOTE SULL'AUTORE

Mi chiamo Roberto Premoli ed ho cominciato ad interessarmi a GNU/Lnux dal 1999.
Sono contattabile agli indirizzi r00(chiocciola)excite.it oppure roberto.premoli(chiocciola)tiscali.it

RINGRAZIAMENTI

Varie sono state le componenti che hanno contribuito alla realizzazione di questo scritto.

kahuna, un frequentatore assiduo della chat presente sul server irc.oltrelinux.com, canale #oltrelinux che mi ha fornito una prima bozza incompleta della traduzione italiana dell'how-to di openMosix per redhat, la quale mi ha causato la frustrazione di non avere disponibile un how-to per debian, e il conseguente desiderio di scriverlo :-). Kahuna mi scusera' se lo cito solo per nick, ma non mi ricordo il suo vero nome. Chi vuole contattarlo, lo cerchi in chat.

Un "grazie" speciale alla **determinante** collaborazione di Fabrizio G. Ficca, reperibile all'indirizzo fab@openlabs.it. Da solo ero riuscito ad implementare un cluster di due nodi didattico, Fabrizio mi ha aiutato a implementare un cluster di 4 nodi di cui 3 con una scheda compact flash (quelle delle macchine fotografiche digitali) al posto dell'hard disk (la configurazione flash l'ha fatta tutta lui): inoltre, mi e' stato di supporto per la parte di configurazione delle connessioni di rete.
Il cluster e' stato presentato al webbit 2003 con notevole riscontro da parte degli spettatori. (si veda www.openlabs.it per ulteriori dettagli)
Fabrizio, e' anche, al momento in cui scrivo, il segretario della associazione culturale www.openlabs.it, attiva sul territorio di Milano e Lombardia, con corsi su GNU/Linux ed altre iniziative tese alla diffusione della filosofia opensource: vi invito a consultare il sito per avere ulteriori informazioni.
Openlabs e' sempre a disposizione di chiunque necessiti supporto in ambito opensource e accetta l'apporto di tutti coloro che vogliono contribuire.