ISTITUTO NAZIONALE DI FISICA NUCLEARE

Sezione di Bari

INFN/TC-03/04
21 Marzo 2003

INSTALLAZIONE, CONFIGURAZIONE E TEST DI UN CLUSTER DI PC PER CALCOLO PARALLELO E SERIALE CON SOFTWARE OPEN-SOURCE

Domenico Diacono

INFN-Sezione di Bari

Sommario

Questo documento illustra in modo dettagliato i passi necessari all'installazione di un cluster di PC Linux con il pacchetto software OSCAR. Il cluster è dotato di bilanciamento automatico del carico mediante openMosix e di ambiente di sviluppo parallelo MPI/PVM. Nel documento si dimostra come anche l'ambiente di calcolo parallelo può trarre beneficio della presenza del bilanciamento automatico del carico.

Published by ***SIS–Pubblicazioni***
Laboratori Nazionali di Frascati

INDICE

1 INTRODUZIONE

Nel presente documento sono fornite indicazioni per la realizzazione di un cluster di PC da utilizzare principalmente per il calcolo parallelo, capace di gestire in modo efficiente anche il calcolo seriale.

L'utente che ha la necessità di eseguire un programma si collega ad una sola macchina, il "gateway", dove risiede la sua directory personale, e lancia il suo programma come farebbe su un PC singolo: l'uso del cluster non impone la revisione del codice. Il sistema di bilanciamento infatti si occupa di distribuire i programmi alle macchine "client" scegliendo dinamicamente la macchina con minore carico. L'utente può anche decidere di modificare i propri programmi per utilizzare le librerie parallele MPI/PVM. In questo caso i test delle prestazioni mostrano che la presenza di openMosix permette un migliore sfruttamento delle risorse di calcolo disponibili.

Questa guida non intende in alcun modo sostituirsi alle guide e manuali d'installazione dei pacchetti software utilizzati, dei quali costituisce un complemento.

Nel seguito ai comandi da digitare direttamente nella shell Linux verrà premesso il simbolo `#`.

1.1 Hardware

L'hardware utilizzato per l'installazione e per i test consiste in un insieme di 5 server, ciascuno così configurato:

- **MotherBoard**: APPRO 1124 Tyan Thunder K7 S2462 ung – Chipset AMD 760 MP
- **Processore**: 2x Athlon MP 2000+
- **RAM:** 1Gb DDR ECC REG.
- **HDD**: 18.1 Gb SCSI Ultra 160 – 10.000 rpm
- **Controller**: EIDE UDMA 100
- **SVGA:** Ati 8MB
- **CDROM**: EIDE Slim
- **NIC:** 2x 10/100 3COM 3C920
- Seriale+Parallela +2USB+2PS/2

Le macchine sono fisicamente collocate in un rack, insieme con uno switch KVM Master View Pro che permette all'operatore di interagire con le macchine usando un solo monitor e una sola tastiera, e ad uno switch ethernet 10/100 Allied Telesyn FS 716.

La prima macchina, il "gateway", ha una interfaccia di rete collegata verso l'esterno e quindi configurata con indirizzo pubblico, e l'altra interfaccia collegata allo switch ethernet e configurata con indirizzo di rete privata. Tutte le altre macchine hanno una sola scheda di rete attiva collegata allo switch e configurata con indirizzo privato. Lo schema di collegamento, che è chiamato "beowulf", è illustrato nella fig. 1, dove con le linee tratteggiate sono indicati i collegamenti ethernet:

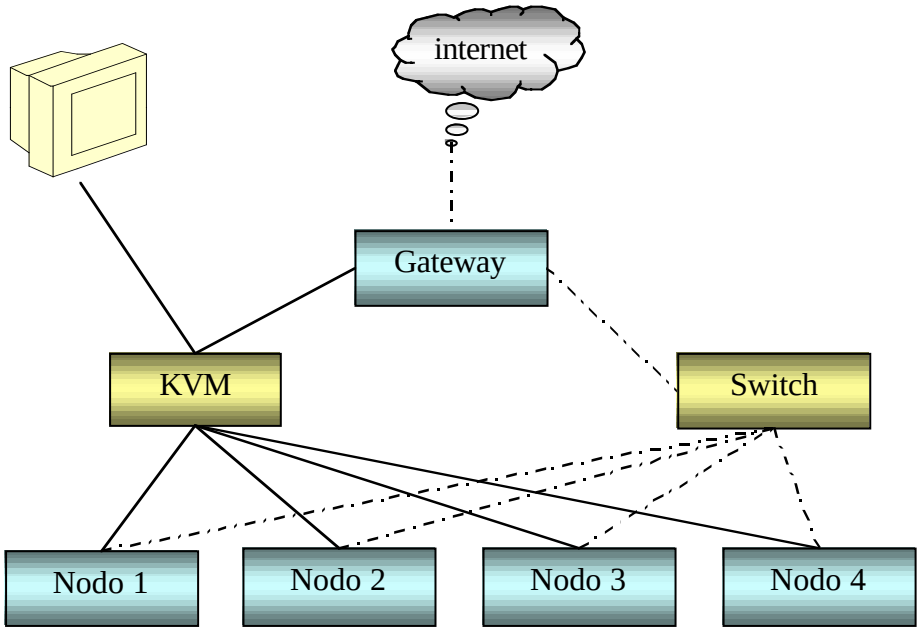


Fig. 1: Schema di collegamento del cluster Beowulf

1.2 Software

Per l'installazione del software sui nodi del cluster è stato utilizzato il sistema OSCAR, liberamente disponibile all'indirizzo http://oscar.sourceforge.net. Si tratta di un pacchetto che al suo interno integra dei software di installazione e gestione per cluster di PC. Brevemente l'installazione procede come segue: innanzitutto si installa il server principale; su di esso si crea una copia, detta "immagine", del file system dei client; infine si installa tale file system su ogni client. I software presenti in OSCAR si occupano di sincronizzare gli account degli utenti su tutte le macchine del cluster, e di rendere possibile la connessione tramite ssh tra i nodi senza la ripetizione della password (autenticazione RSA). La versione utilizzata è la 2.1.

Del bilanciamento automatico del carico si occupa il software openMosix, disponibile all'indirizzo http://openmosix.sourceforge.net. OpenMosix è una estensione del kernel Linux standard che, una volta applicata, permette al cluster di ottimizzare lo sfruttamento delle risorse tramite i meccanismi di load-balancing dinamico e di ottimizzazione dell'utilizzo della memoria. I programmi in esecuzione vengono divisi in due parti: una residente sul sistema di partenza, ed un'altra capace di muoversi nel cluster alla ricerca della macchina con più risorse disponibili. L'utente non ha bisogno di conoscere la topologia del cluster, che può anche variare nel tempo. La versione utilizzata è la 2.4.18-4.

Per la verifica delle prestazioni del cluster sono stati usati i benchmark paralleli NAS (NASA Advanced Supercomputing Division): si tratta di un insieme di 8 programmi sviluppati per permettere di misurare le prestazioni di supercomputer paralleli. I benchmarks, derivati da applicazioni di fluido dinamica computazionale (CFD), consistono di cinque applicazioni kernel e tre pseudo-applicazioni. La versione utilizzata, liberamente disponibile all'indirizzo http://www.nas.nasa.gov/NAS/NPB/, è la 2.4.

2 INSTALLAZIONE DEL GATEWAY

2.1 Sistema operativo

La distribuzione utilizzata per l'installazione del sistema operativo Linux sul gateway è la RedHat 7.3; durante la fase di installazione è sufficiente accettare i default proposti, con le seguenti precauzioni:

- Scegliere il tipo di installazione "workstation". E' il caso di eliminare, usando la lista dei pacchetti, tutto quanto è inutile per lo scopo per il quale sarà impiegato il cluster, e di aggiungere invece i pacchetti per il "software development".
- Date le successive necessità è opportuno suddividere il disco nelle partizioni `/boot` e `swap` con dimensione di default, `/tftpboot` di circa 2,5 Gb, `/` di circa 3 Gb e `/home` che occupa tutto lo spazio rimanente.
- Le due interfacce di rete devono essere configurate in modo che eth0 sia connessa alla rete privata[1] ed eth1 sia connessa alla rete pubblica.
- E' sufficiente impostare il tipo di firewall medio: OSCAR include un suo compilatore di firewall che durante l'installazione imposta correttamente le protezioni.
- Per evitare problemi di compatibilità tra software localizzati è necessario installare la versione en_US: in caso contrario l'installazione del cluster può fallire.

Ad installazione ultimata, se è necessario modificare le impostazioni delle interfacce di rete, è possibile usare il comando

```
# neat
```

altrimenti i file da modificare manualmente sono `/etc/resolv.conf` per il DNS, `/etc/sysconfig/network-scripts/ifcfg-eth0` ed `ifcfg-eth1` per le interfacce, `/etc/sysconfig/network` per il gateway e il nome host.

I pacchetti software obsoleti si possono aggiornare tramite l'utility "up2date", che è attivata da un applet del barra di controllo. Bisogna però avere l'avvertenza di lanciare innanzitutto

```
# up2date –-configure
```

e scegliere l'opzione "After installation, keep binary packages on disk". I pacchetti serviranno in seguito per l'aggiornamento dei client. Con up2date si può aggiornare anche il kernel all'ultima versione supportata da RedHat per la distribuzione 7.3: per la installazione qui discussa è stata usata la versione 2.4.18-18.

Infine è necessario procurarsi la versione 3.5 di openSSH, o comunque l'ultima versione, disponibile all'indirizzo http://www.openssh.org, aggiornare il server e conservare anche questi pacchetti rpm per l'aggiornamento dei client.

2.2 Pacchetti RPM per i client

L'installazione delle macchine client del cluster all'interno di OSCAR è gestita dal

[1] Si ricorda che sono disponibili tre intervalli di indirizzi IP: da 10.0.0.0 a 10.255.255.255; da 172.16.0.0 a 172.32.255.255; da 192.168.0.0 a 192.168.255.255. Per il server non è possibile utilizzare, pena il fallimento della installazione di OSCAR, gli indirizzi 10.0.0.0, 172.16.0.0, 192.168.0.0.

pacchetto “System Installation Suite” (http://www.sisuite.org/): tutti i client vengono installati a partire da una immagine del loro file system costruita e residente sul gateway. Bisogna creare la directory `/tftpboot/rpm`, e affinché sia possibile la costruzione del file system immagine con tutti i pacchetti necessari è necessario copiare all’interno i file `.rpm` della distribuzione standard di Linux, con il comando:

```
# cp /mnt/cdrom/RedHat/RPMS/*.rpm /tftpboot/rpm
```

All’interno della stessa directory bisogna copiare tutti i file RPM di aggiornamento dei pacchetti standard, ovvero tutti i file prelevati in precedenza da up2date, che si trovano in `/var/spool/up2date`, e i file di openSSH. Non vanno invece copiati in tale directory gli aggiornamenti del kernel.

2.3 Installazione di openMosix

Per la distribuzione RedHat 7.3 sono disponibili i pacchetti RPM per l’installazione di openMosix, uno per il kernel (“openmosix-kernel-smp-2.4.18-openmosix4.athlon.rpm”) e l’altro per tutti i programmi utente necessari per sfruttare le caratteristiche del kernel (“openmosix-tools-0.2.4-1.i386.rpm”)

Terminata l’installazione degli rpm bisogna modificare il file `/etc/mosix.map`:

```
# MOSIX-#  IP  number-of-nodes
# ============================
1   192.168.0.254           1
1   <indirizzo pubblico>ALIAS
2   192.168.0.2             4
```

dove ovviamente bisogna sostituire ad <indirizzo pubblico> l’IP della macchina raggiungibile da internet. Per il corretto funzionamento di openMosix può essere inoltre necessario modificare come segue il file `/etc/hosts`:

```
192.168.0.254    <nome.dominio>           <nome>
127.0.0.1        localhost.localdomain    localhost
```

dove <nome.dominio> è il nome completo del cluster. La configurazione, se si usa il boot-loader GRUB, è terminata, altrimenti nel file `/etc/lilo.conf` bisogna aggiungere il kernel openMosix:

```
image=/boot/vmlinuz-2.4.18-openmosix4smp
        label=2.4.18-om4smp
        read-only
        initrd=/boot/initrd-2.4.18-openmosix4smp.img
```

Dopo aver creato la directory `/mfs`, nel file `/etc/fstab` deve essere aggiunta la riga

seguente:

```
mfs_mnt      /mfsmfs      dfsa=1       0    0
```

In questo modo sarà abilitato il Mosix File System: è un file system caratteristico di openMosix che permette di accedere ai file system remoti nel cluster come se fossero montati localmente. DFSA è l'acronimo di Direct File System Access ed è un'ulteriore ottimizzazione che permette a processi remoti di eseguire delle chiamate di sistema sui file localmente invece di rispedirle indietro al nodo di partenza del processo.

Al termine si avrà una macchina capace di partire sia con il kernel openMosix che con l'ultimo kernel disponibile per la distribuzione RedHat.

3 INSTALLAZIONE DEL CLUSTER

3.1 Preparazione

L'installazione dei client, nonché di tutti i pacchetti necessari al gateway per il funzionamento all'interno del cluster, è gestita da OSCAR.

Nel caso qui discusso è stato sufficiente seguire punto per punto la guida di OSCAR, consultabile all'indirizzo: http://oscar.sourceforge.net/docs/oscar_installation_2.1.pdf, con le modifiche necessarie a far si che il sistema finale potesse configurarsi anche come cluster openMosix.

Prima di iniziare bisogna innanzitutto accertarsi che la configurazione di rete del gateway sia quella voluta: dopo aver fatto ripartire il server, si esegue il comando

```
# ifconfig –a
```

e si esamina la prima parola della terza riga nell'output per ciascuna scheda di rete: se non è UP prima di procedere è necessario risolvere il problema.

Sistemata la configurazione di rete del server se si vuole utilizzare il tool GANGLIA (http://ganglia.sourceforge.net/) per il controllo del cluster a distanza, bisogna procedere come segue: supponendo che si sia decompresso il pacchetto OSCAR nella directory `/root` si fa partire l'installazione con il comando

```
# cd /root/oscar2.1
# ./install_cluster eth0
```

dove eth0 è la porta ethernet del gateway connessa alla rete privata. Si interrompe l'installazione quando appare l'interfaccia grafica e si esegue:

```
# export OSCAR_HOME='/root/oscar2.1'
#./scripts/opd
```

In questo modo vengono installati tutti i moduli Perl necessari all'OSCAR Package Downloader (OPD). All'interno di OPD si seleziona il server e il pacchetto "Ganglia" da scaricare.

3.2 Installazione dei client

A questo punto è possibile finalmente far partire l'installazione con:

```
# ./install_cluster eth0
```

E' assolutamente necessario seguire i passi che sono mostrati nell'interfaccia grafica nell'ordine in cui appaiono. Fino al terzo passo ("*Install OSCAR server packages*") si può procedere seguendo le istruzioni della guida all'installazione. Prima del quarto passo ("*Build OSCAR client image*") bisogna modificare il file di partizionamento dei dischi client.

Il quarto passo, infatti, costruisce, a partire dai pacchetti copiati in precedenza nella directory `/tftpboot/rpm`, l'immagine del file system dei client e specifica lo schema delle partizioni nelle quali verranno divisi i dischi dei client. Per semplicità d'ora in poi ci riferiremo alla directory dove risiede l'immagine, ossia

```
/var/lib/systemimager/images/oscarimage/
```

come alla directory `$IMAGE`. Prima di avviare la costruzione dell'immagine è quindi necessario adattare il file che specifica il partizionamento dei dischi dei client alle proprie esigenze. Nel caso in esame il disco da 18 Gbyte è stato suddiviso in 4 partizioni modificando il file `/root/oscar2.1/oscarsample/sample.disk.scsi` come segue:

```
/dev/sda1  47      ext3/boot        defaults
/dev/sda3  1992    ext3/            defaults
/dev/sda4  1992    swap
/dev/sda2  13477   ext3/scratch     defaults
nfs_oscar:/home -  nfs      /home        rw
```

Come si vede ogni client ha un'area di lavoro (riga 4), che metterà in seguito a disposizione degli utenti tramite il Mosix File System (MFS); la "home" di ciascun utente risiede solo sul gateway, e i client vi accedono tramite NFS (riga 5).

Subito dopo aver costruito l'immagine, e prima di passare al quinto punto ("*Define OSCAR clients*") ci si può dedicare all'aggiornamento del kernel. Dopo aver copiato in `$IMAGE/tmp` gli rpm necessari, ossia quello scaricato in precedenza da `up2date` e quello preso dal sito di openMosix, ci si porta nella directory `$IMAGE` e si esegue il comando

```
# chroot . sh
```

A questo punto è possibile installare l'rpm con il comando standard:

```
# rpm –ivh <nome file>
```

Perché i nuovi kernel siano correttamente installati nei client bisogna modificare il file `$IMAGE/etc/systemconfig/systemconfig.conf`: per ogni kernel si aggiunge una nuova sezione, in modo che alla fine il file appaia come nell'esempio:

```
CONFIGBOOT = YES
CONFIGRD = YES

[BOOT]
```

```
ROOTDEV = /dev/sda3
BOOTDEV = /dev/sda
DEFAULTBOOT = 2.4.18-om4smp
APPEND = noapic

[KERNEL0]
PATH=/boot/vmlinuz-2.4.18-openmosix-4smp
LABEL = 2.4.18-om4smp

[KERNEL1]
PATH=/boot/vmlinuz-2.4.18-18.7.xsmp
LABEL = 2.4.18-18
```

Si fa notare che sebbene il kernel sia contrassegnato dalla versione 2.4.18-18.7.xsmp, la label usata è semplicemente 2.4.18-18, così come la label del kernel openMosix è stata abbreviata da 2.4.18-openmosix4smp ad 2.4.18-om4smp. La ragione è nel fatto che l'esecuzione di `/sbin/lilo` sui client al termine della installazione fallisce se si usano label troppo lunghe.

E' anche necessario creare nella directory `$IMAGE/lib/modules` un link che permetta al programma che crea il RAM disk al termine della installazione via rete dei client di trovare i moduli per il kernel 2.4.18-18.7.xsmp, che viene riconosciuto come 2.4.18-18:

```
# ln –s 2.4.18-18.7.xsmp 2.4.18-18
```

Lo script "*kernel_picker*" fornito da OSCAR dovrebbe eseguire le stesse operazioni qui descritte: per i motivi illustrati, però, lo script certamente non può produrre, in casi simili a quello esaminato, un aggiornamento valido.

Per il funzionamento del cluster è necessario installare nella immagine anche l'RPM degli userland-tools, e questo è un buon momento per farlo: i passi da seguire sono gli stessi illustrati per l'installazione dell'aggiornamento dei kernel. Infine per terminare la configurazione della parte openMosix del cluster è necessario copiare il file `/etc/mosix.map` nella corrispondente posizione della immagine `$IMAGE/etc`, in modo che a costruzione ultimata sia presente identico in tutte le macchine.

Come mostrato in precedenza, le macchine sono tutte dotate di un'area del disco locale disponibile per gli utenti: per renderla accessibile occorre abilitare su tutti i nodi il Mosix File Sistem. Naturalmente queste aree non saranno disponibili nel caso in cui si farà partire il cluster nella configurazione senza openMosix. Per realizzare tale condivisione deve essere creata la directory `$IMAGE/mfs` e la stessa riga che è stata aggiunta in `/etc/fstab` nel gateway va aggiunta nel file `$IMAGE/etc/fstab`:

```
mfs_mnt      /mfsmfs      dfsa=1       0   0
```

Dopo aver fatto ripartire l'intero sistema è consigliabile, per rendere più agevole l'accesso,

creare dei link simbolici sia sul gateway:

```
#ln –s /mfs/2/scratch    /sa2
```

che sui client, in modo che le applicazioni in qualsiasi nodo stiano girando possano sfruttare l'accesso diretto alle aree scratch condivise:

```
# cexec "ln –s /mfs/2/scratch    /sa2"
```

La definizione dei client e il successivo setup della rete ai punti cinque e sei dell'installazione non presentano difficoltà. Per rendere omogenea la numerazione dei nodi tra openMosix ed il sistema di controllo del cluster C3 può essere comodo numerare i client a partire dal numero 2, assegnare gli indirizzi a partire dal 192.168.0.2 ed aggiungere nel file /etc/c3.conf una riga per spostare a 2 l'indice corrispondente al primo nodo di calcolo. La prima installazione dei nodi avviene via rete, e quindi bisogna selezionare sui client il boot via PXE o via dischetto con Etherboot: in questo secondo caso il floppy di boot può essere creato da OSCAR. Terminata la fase di download la macchina deve ripartire dal suo disco interno, e quindi bisogna modificare la sequenza di avvio nel BIOS.

3.3 Aggiornamento dei client

Nel caso in cui si renda necessario aggiornare un pacchetto software o il kernel stesso, una volta installati tutti i nodi, è possibile adottare la procedura descritta in questo paragrafo: ad esempio può essere necessario applicare il kernel openMosix su un cluster già interamente costruito utilizzando l'installazione OSCAR standard. L'aggiornamento del gateway non comporta particolari difficoltà; presenta qualche particolarità invece l'aggiornamento dei client e l'aggiornamento dell'immagine, questo necessario a far sì che eventuali nuovi client entrino senza problemi nel cluster.

3.3.1 Aggiornamento delle macchine già operative

In aiuto dell'amministratore OSCAR mette a disposizione il sistema di controllo Cluster Command and Control (http://www.csm.ornl.gov/torc/C3/), già citato ed utilizzato in precedenza, che permette l'esecuzione contemporanea su tutti i client di comandi amministrativi. Grazie a questo software i passi da seguire sono i seguenti:

- Copiare i file rpm in un punto visibile da tutti i client, ad esempio si può creare la directory `/home/mosix` sul gateway.
- Installare l'aggiornamento del kernel e gli userland-tools su tutti i client con i comandi:
  ```
  # cexec "rpm –ivh /home/mosix/openmosix-kernel-smp-2.4.18-openmosix4.athlon.rpm
  # cexec "rpm –ivh /home/mosix/openmosix-tools-0.2.4-1.i386.rpm"
  ```
- Sincronizzare le mosix.map
  ```
  # cpush /etc/mosix.map /etc/mosix.map
  ```
- Installare nel bootloader del client il nuovo kernel, prelevando da una macchina il file `/etc/lilo.conf`, modificandolo con l'aggiunta del riferimento al nuovo kernel

(come già visto nel paragrafo 2.3), spostando su tutti i client il file modificato ed infine eseguendo `/sbin/lilo` su ogni nodo:

```
# cget :2 /etc/lilo.conf
# vi lilo.conf_oscar_cluster_nodo2.thclus
# cpush lilo.conf_oscar_cluster_nodo2.thclus
/etc/lilo.conf
# cexec /sbin/lilo
```

- Con un procedimento analogo aggiungere la riga riguardante il file system mfs nel file `/etc/fstab` dei client.
- Infine riavviare tutti i client
  ```
  # cshutdown –r t 0
  ```

3.3.2 Aggiornamento dell'immagine

E' necessario aggiornare l'immagine dei client che risiede sul gateway per far sì che i nuovi nodi entrino a far parte del cluster appena terminata la loro installazione. La procedura da seguire è la seguente, ricordando che sempre dove compare `$IMAGE` bisogna sostituire `/var/lib/systeminager/images/oscarimage`:

- Copiare gli rpm nella directory nella quale risiede l'immagine ed installarli:
  ```
  # cp –a /root/mosix $IMAGE/tmp
  # cd $IMAGE
  # chroot . sh
  # rpm –ivh /tmp/mosix/openmosix-kernel-smp-2.4.18
  openmosix4.athlon.rpm
  # rpm –ivh /home/mosix/openmosix-tools-0.2.4-1.i386.rpm
  ```
- Aggiungere al file `$IMAGE/etc/systemconfig/systemconfig.conf` le righe seguenti:

  ```
  [KERNEL0]
     PATH=/boot/vmlinuz-2.4.18-openmosix4smp
     LABEL = 2.4.18-om4smp
  ```

  e impostare se necessario la label di default per il boot nella voce `DEFAULTBOOT`.
- Tornare nel file system originale del gateway e copiare nella immagine la mappa del cluster opportunamente aggiornata
  ```
  # cp etc/mosix.map/ $IMAGE/etc/
  ```
  Questa mappa aggiornata deve essere identica a quella presente su tutti i nodi già funzionanti.
- Infine aggiornare il file `$IMAGE/etc/fstab` aggiungendo la riga relativa al file system mfs.

4 TEST DELLE PRESTAZIONI

4.1 Installazione e configurazione

I benchmark scelti per verificare le prestazioni di calcolo parallelo del cluster sono i benchmark NAS, introdotti nel paragrafo 1.2, nella versione 2.4; per la compilazione e l'esecuzione è stata usata la implementazione LAM/MPI, versione 6.5.7, delle librerie parallele, e il compilatore g77 versione 2.96. La documentazione su LAM/MPI si trova all'indirizzo http://www.lam-mpi.org/.

Dopo aver estratto i sorgenti nella directory `/usr/local/src/` è necessario adattare l'ambiente di compilazione modificando il file `/usr/local/src/NPB2.4/NPB2.4-MPI/config/make.def`. Per poter compilare i test l'unica variazione necessaria per problemi nella compilazione è stata la sostituzione della routine random predefinita `randi8` con l'altra proposta nel template `randdp`.

Quando si compilano i benchmarks bisogna indicare la classe del test, vale a dire la sua complessità, e il numero di processi nei quali dividere il calcolo, ad esempio:

```
# make mg CLASS=C NPROCS=8
```

I programmi compilati vanno spostati dalla directory originale `/usr/local/src/NPB2.4/NPB2.4-MPI/bin` in una directory che sia visibile da tutte le macchine del cluster e sia disponibile agli utenti.

Per eseguire i benchmarks nell'ambiente LAM/MPI è necessario innanzitutto fare il login con un utente normale non-root: per questioni di sicurezza non è concesso all'utente root l'uso delle risorse parallele. Si può impostare un ambiente comune a tutti gli utenti descrivendolo nel file `/opt/lam-6.5.7/etc/lam-bhost.def`:

```
thclus.ba.infn.it       cpu=2
node2.thclus            cpu=2
node3.thclus            cpu=2
node4.thclus            cpu=2
node5.thclus            cpu=2
```

Ciò non toglie che ogni utente nella home directory può definirsi il suo ambiente di esecuzione nel file `lamhosts`.

Per far partire l'ambiente d'esecuzione parallelo si usa il comando :

```
# lamboot –v
```

quindi per eseguire i test:

```
# mpirun –c <numero dei processi> –v <nome-programma>
```

Il numero di copie di programma usato (numero dei processi) deve coincidere con il numero di processi per i quali il test è stato compilato: ad esempio

```
# mpirun –np 16 –v mg.C.16
```

4.2 Risultati

I test della suite utilizzati sono MG, LU, BT, SP, IS, EP, CG. Tutti i test hanno un timer, e al termine dell'esecuzione si riesce agevolmente a sapere quanto tempo il sistema ha impiegato nel calcolo, i Megaflops/sec totali e i Megaflops/sec per processo. Questi valori permettono di confrontare le prestazioni dello stesso cluster al variare di un parametro della configurazione, ad esempio la presenza o meno del kernel con supporto openMosix.

TAB. 1: Risultati benchmark LU di classe B

Processi	Mflops/sec	Mflops/sec/nodo	Tempo (sec)
4	495	99,0	1008
8	941	188,2	530
16	709	141,8	704
32	251	50,2	1984
64	96	19,2	5197

TAB. 2: Risultati benchmark LU di classe B con openMosix

Processi	Mflops/sec	Mflops/sec/nodo	Tempo (sec)	Differenza (%)
4	491	98,2	1015	+0,8
8	948	189,6	526	-0,7
16	958	191,6	521	-26,0
32	858	171,6	581	-70,7
64	684	136,8	730	-86,0

Le tabelle 1 e 2 mostrano i risultati del benchmark LU condotto da un singolo utente su un cluster di 5 server biprocessore, rispettivamente con kernel 2.4.18-18smp e 2.4.18-openmosix4smp. La differenza si riferisce al tempo impiegato. LU è un kernel benchmark che usa il metodo SSOR (symmetric successive over relaxation) per risolvere un sistema diagonale a sette blocchi risultante dalla discretizzazione alle differenze finite delle equazioni di NavierStokes in 3 dimensioni.

Se il numero dei processi non supera il numero dei processori, in questo caso dieci, le differenze tra le due configurazioni sono trascurabili. Viceversa se il numero dei processi supera quello dei processori disponibili appare evidente l'opera di ottimizzazione dello sfruttamento delle risorse attuata dal kernel openMosix. L'aumento della velocità infatti nel caso di 64 processi arriva all'86%. Ovviamente non è utile in genere, nella pratica, compilare ed eseguire un programma MPI con un numero di processi superiore a quello dei processori disponibili: quest'espediente è servito per verificare il comportamento dei due kernel in presenza di un forte carico di processi contemporanei.

TAB. 3: Risultati benchmark classe B

Test	Num. processi	Mflops/sec	Mflops/sec/ nodo	Tempo (sec)
MG	128	9	1,8	2097
BT	36	377	75,4	1862
EP	64	30	6,0	72
IS	64	5	1,0	73
CG	64	11	2,2	4907
SP	36	120	24,0	2953

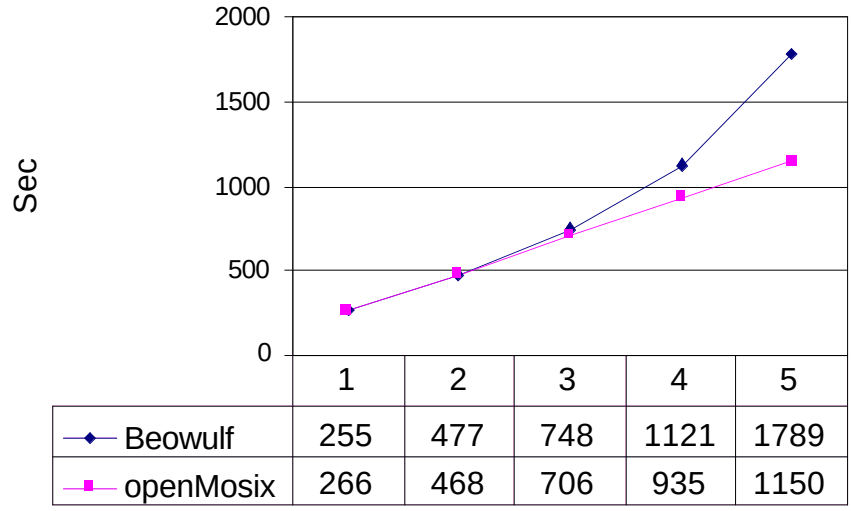
TAB. 4: Risultati benchmark classe B con openMosix

Test	Num. Processi	Mflops/sec	Mflops/sec/ nodo	Tempo (sec)	Differenza (%)
MG	128	62	12,4	312	-85
BT	36	1009	201,8	696	-63
EP	64	33	6,6	65	-10
IS	64	9	1,8	39	-47
CG	64	62	12,4	879	-82
SP	36	486	97,2	687	-77

Nelle tabelle 3 e 4 sono riportati i risultati dei diversi benchmark eseguiti con numero di processi più grande del numero dei processori: come si vede le differenze percentuali variano al variare del benchmark utilizzato, ma si mantengono in ogni caso rilevanti. Risultati molto simili si ottengono per i test di classe C con complessità superiore.

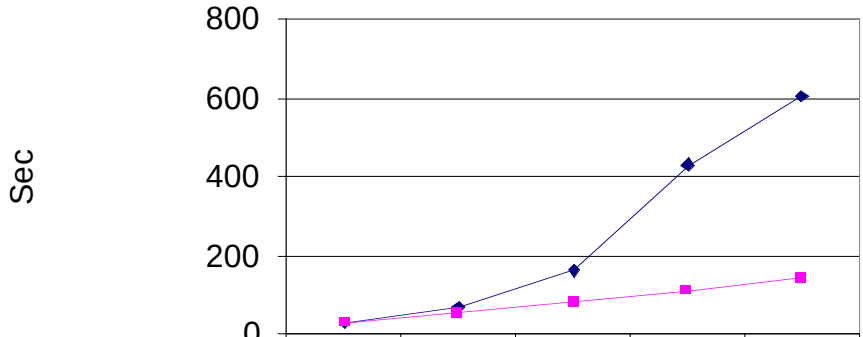
Un modo più aderente al reale utilizzo per provare le prestazioni del sistema nelle due configurazioni possibili è quello di far eseguire contemporaneamente da diversi utenti dei programmi MPI compilati per un numero di processi inferiore a quello dei processori.

La figura 2 mostra il tempo d'esecuzione di un singolo test LU, classe B, compilato per 8 processi ed eseguito contemporaneamente da un numero di utenti variabile da 1 a 5; la figura 3 mostra il tempo di esecuzione del test MG di classe B, compilato per 8 processori, nelle stesse condizioni. Il test MG usa un metodo Vcycle MultiGrid per calcolare la soluzione della equazione scalare 3D di Poisson.



	1	2	3	4	5
Beowulf	255	477	748	1121	1789
openMosix	266	468	706	935	1150

FIG. 2: Tempo medio di esecuzione LU-B-8.



	1	2	3	4	5
Beowulf	31	71	162	429	603
openMosix	32	54	81	110	144

Fig 3: Tempo medio di esecuzione MG-B-8

5 CONCLUSIONI

Nel documento si è illustrata una procedura di installazione rapida di un cluster Beowulf, che combinando le esistenti tecnologie open-source realizza un sistema multiutente capace di rispondere alle esigenze di calcolo seriale e parallelo.

Il sistema si presta all'utilizzo come cluster di calcolo anche per chi non utilizza MPI/PVM grazie alla presenza di openMosix: l'utente può collegarsi al gateway ed eseguire i suoi programmi senza conoscere la reale architettura fisica del cluster. OpenMosix si occupa di spostare la parte "utente" del processo in esecuzione verso i nodi meno carichi, ottimizzando le prestazioni e l'utilizzo delle risorse di calcolo.

Sono stati anche condotti dei test delle prestazioni, dai quali è risultato evidente che le applicazioni di calcolo parallelo con MPI beneficiano della presenza di openMosix, sfruttando in modo più efficiente la potenza di calcolo a disposizione. Da notare che i processi MPI nella configurazione descritta non possono migrare, e che il rapporto tra la velocità di esecuzione con o senza openMosix varia al variare del benchmark usato.

Una esatta analisi dei meccanismi di ottimizzazione di openMosix esula però dallo scopo di questo documento.

6 RINGRAZIAMENTI

Si ringraziano il Dott. Pietro Colangelo e la Sig.ra Bruna Tataranni per i consigli dati durante la stesura di questo documento.

7 RIFERIMENTI

In aggiunta ai riferimenti dati nel corso del documento si segnalano i seguenti link:

1 http://www.na.infn.it/compreso/mosix/mosix-slides : presentazione su Mosix con alcuni test di performance.
2 http://howto.ipng.be/openMosix-HOWTO/: the openMosix How-To.
3 http://www.plogic.com/bps : The Beowulf performance suite.
4 http://www.cacr.caltech.edu/beowulf/tutorial/building.html Building a Beowulf system.
5 http://www.phy.duke.edu/brahma/beowulf_online_book Robert G. Brown Engineering a Beowulf style Compute Cluster.
6 http://aggregate.org/PPLINUX/19980105/pphowto.html LINUX Parallel Processing How-To