

# Creare un CLUSTER

## Premessa

Dopo aver seguito questa guida sarete in grado far girare i vostri programmi **MPI** in un cluster composto da 2 o più computer, maggiori info qui [https://it.wikipedia.org/wiki/Computer\\_cluster](https://it.wikipedia.org/wiki/Computer_cluster).

Le prove sono state effettuate con **UBUNTU 18.04** installato su partizione dedicata per evitare colli di bottiglia derivanti dalle Virtual-machine.

**NB: TUTTI I COMANDI VANNO ESEGUITI SENZA GLI APICI “ ”.**

## Prerequisiti

Cosa fondamentale è avere installato la stessa, e magari l'ultima, versione disponibile di MPI su tutte le macchine che comporranno il cluster, per fare questo suggerisco il classico “**sudo apt-get install mpich**”.

### 0) Configurare il file HOSTS

Questo è un passaggio facoltativo ma, se seguito, consentirà di risparmiare del tempo successivamente.

Ogni computer in rete ha un indirizzo IP che lo identifica univocamente. Quello che stiamo per andare a fare è associare un nome, anch'esso univoco, all'IP di master e slave.

A questo punto bisogna installare i net-tools che conterranno degli strumenti che ci serviranno in futuro, quindi digitiamo “**sudo apt-get install net-tools**”.

Per conoscere l'IP locale, una volta installati i net-tools, basta digitare : “**ifconfig**”.

Per visualizzare il file da modificare basta digitare “**cat /etc/hosts**”.

Per modificare il file digitiamo “**sudo -i gedit /etc/hosts**”, è preferibile farlo su tutti i computer.

Il vostro file hosts dovrà essere strutturato nel seguente modo :

NOME1 192.0.0.0

NOME2 192.0.0.0

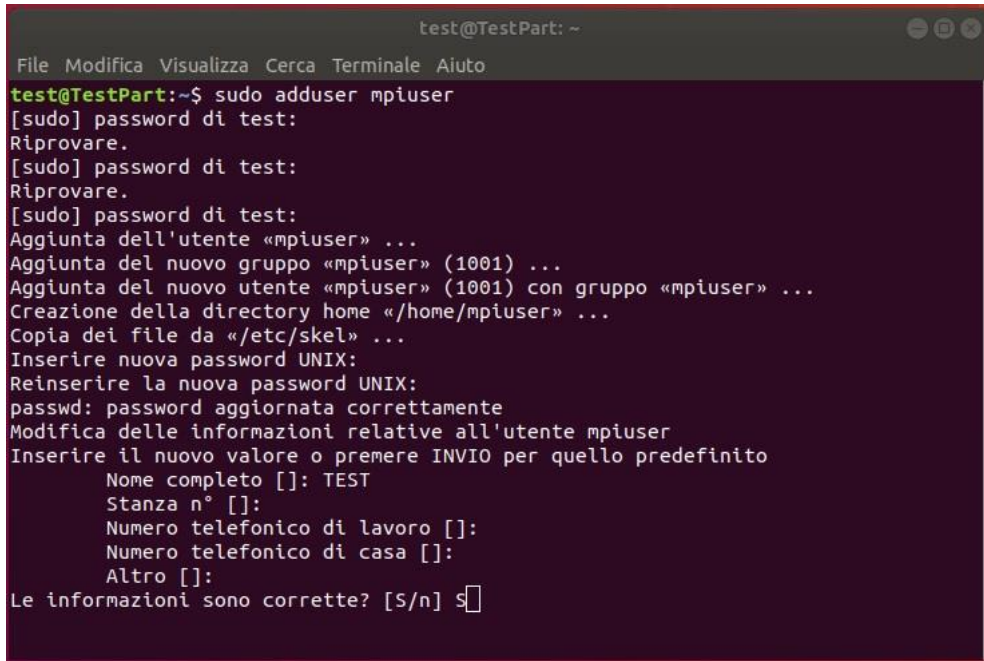
Da sostituire con il nome desiderato e l'IP del rispettivo computer.

Una volta fatto questo per verificare il successo dell'operazione basta digitare (ipotizziamo dal master) “**ping slave**” e vedere se esiste oppure se lo slave risulta “**unreachable**”, in quest'ultimo caso ricontrolliamo i passaggi precedenti. Con il comando “**ping 192.0.0.0**” (dove 192.0.0.0 è l'IP di un altro pc presente in rete) è possibile vedere anche la **latenza**, dato molto importante per un cluster.

## 1) Creare un nuovo utente

Dal terminale del nostro master digitiamo **"sudo adduser mpiuser"**. Ci verranno richiesti i nuovi dati dell'account tra cui la nuova password. Possiamo saltare l'aggiunta di informazioni premendo semplicemente invio e lasciando lo spazio vuoto, ovviamente non è possibile farlo per la password.

Testiamo il nuovo account digitando **"su mpiuser"**, inseriamo la password e verifichiamone il corretto funzionamento. A questo punto per fare il logout e ritornare al nostro account principale digitiamo **"exit"**.

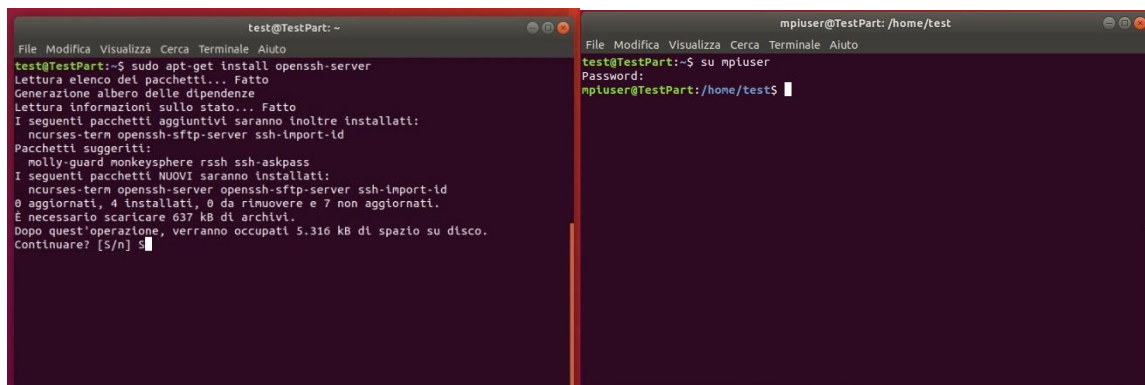


```
test@TestPart: ~  
File Modifica Visualizza Cerca Terminale Aiuto  
test@TestPart:~$ sudo adduser mpiuser  
[sudo] password di test:  
Riprovare.  
[sudo] password di test:  
Riprovare.  
[sudo] password di test:  
Aggiunta dell'utente «mpiuser» ...  
Aggiunta del nuovo gruppo «mpiuser» (1001) ...  
Aggiunta del nuovo utente «mpiuser» (1001) con gruppo «mpiuser» ...  
Creazione della directory home «/home/mpiuser» ...  
Copia dei file da «/etc/skel» ...  
Inserire nuova password UNIX:  
Reinserire la nuova password UNIX:  
passwd: password aggiornata correttamente  
Modifica delle informazioni relative all'utente mpiuser  
Inserire il nuovo valore o premere INVIO per quello predefinito  
Nome completo []: TEST  
Stanza n° []:  
Numero telefonico di lavoro []:  
Numero telefonico di casa []:  
Altro []:  
Le informazioni sono corrette? [S/n] S
```

## 2) Configurare la connessione SSH

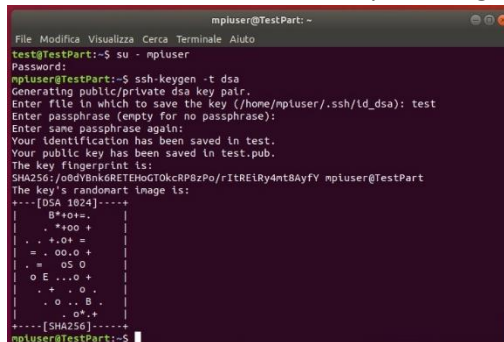
Dal nostro master digitiamo **"sudo apt-get install openssh-server"**.

Una volta terminato, su UBUNTU 18.04 dovrebbe già essere installato, facciamo il login al nostro nuovo account con **"su - mpiuser"**.



```
test@TestPart: ~  
File Modifica Visualizza Cerca Terminale Aiuto  
test@TestPart:~$ sudo apt-get install openssh-server  
Lettura elenco dei pacchetti... Fatto  
Generazione albero delle dipendenze  
Lettura informazioni sullo stato... Fatto  
I seguenti pacchetti aggiuntivi saranno inoltre installati:  
ncurses-term openssh-sftp-server ssh-import-id  
Pacchetti suggeriti:  
nolly-guard monkeysphere rsync ssh-askpass  
I seguenti pacchetti NUOVI saranno installati:  
ncurses-term openssh-server openssh-sftp-server ssh-import-id  
0 aggiornati, 4 installati, 0 da rimuovere e 7 non aggiornati.  
È necessario scaricare 637 kB di archivi.  
Dopo quest'operazione, verranno occupati 5.316 kB di spazio su disco.  
Continuare? [S/n] S  
  
mpiuser@TestPart: /home/test  
File Modifica Visualizza Cerca Terminale Aiuto  
test@TestPart:~$ su mpiuser  
Password:  
mpiuser@TestPart: /home/test$
```

Digitiamo **“ssh-keygen -t dsa”**, in questo modo genereremo una chiave di tipo DSA che dovrà essere copiata nel nostro slave, per fare questo digitiamo **“ssh-copy-id slave”** se invece non avete seguito il passaggio facoltativo 0 dobbiamo digitare **“ssh-copy-id 192.0.0.0”** dove **“192.0.0.0”** va sostituito con l’IP del nostro slave. Questo va fatto per tutti gli slaves.



```
File Modifica Visualizza Cerca Terminale Aiuto
test@TestPart:~$ su - mpiuser
mpiuser@TestPart:~$ ssh-keygen -t dsa
Generating public/private dsa pair.
Enter file in which to save the key (/home/mpiuser/.ssh/id_dsa): test
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in test.
Your public key has been saved in test.pub.
The key fingerprint is:
SHA256:06dYBnk6RETEHoGTOkcrPBzPo/rItRELRy4nt8AyfY mpiuser@TestPart
The key's randomart image is:
+--[ DSA 1024 ]-----+
|      B*+0+==      |
|      +100 +      |
|      +..0+ =      |
|      +..00.0 +      |
|      +..05 0      |
|      +..0 +      |
|      +..0 +      |
|      +..0 +      |
|      +..0 +      |
|      +..0 +      |
+-----[SHA256]-----+
mpiuser@TestPart:~$
```

Adesso non ci resta altro che abilitare il login senza password, digitiamo :  
**“eval ‘ssh-agent’”** e **“ssh-add ~/.ssh/id\_dsa”**

Adesso testiamo il corretto funzionamento, sempre dal vostro account mpiuser, digitiamo **“ssh slave”**.

Se la connessione SSH è avvenuta con successo dovremmo vedere il nostro terminale loggato con l’account dello slave.

### 3) Creiamo la cartelle condivisa via NFS sul MASTER

Dal nostro master digitiamo **“sudo apt-get install nfs-kernel-server”** per installare il pacchetto.

Una volta terminato (qualora nella vostra distribuzione Linux non fosse già installato), dal nostro master procediamo facendo il login sull’utente che abbiamo creato in precedenza.

Adesso non ci resta che creare la cartella che andremo a condividere, digitiamo dunque:

**“mkdir sharedDir”**.

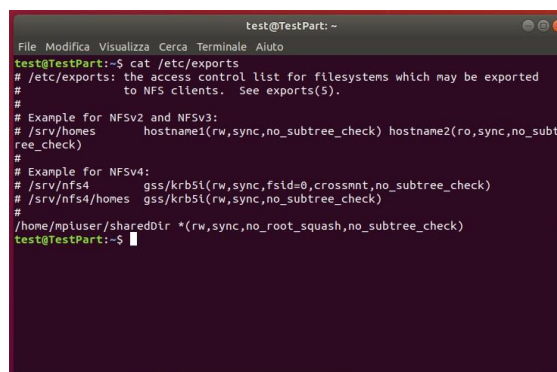
Il passaggio successivo è fondamentale per garantire il tipo di accesso alla cartella, quindi digitiamo:

**“sudo -i gedit /etc/ exports”** ed incolliamo la seguente linea :

**“/home/mpiuser/sharedDir \*(rw,sync,no\_root\_squash,no\_subtree\_check)”**

L’asterisco **“\*”** indica l’indirizzo IP a cui condividere la cartella ( se si lascia l’\* la condivisione viene fatta verso tutti gli slaves ), il testo tra parentesi rappresenta i parametri.

Digitiamo **“cat /etc/exports”** per visualizzare il risultato ottenuto, quindi digitiamo **“exportfs -a”** per rendere i cambiamenti effettuati in exports permanenti.



```
File Modifica Visualizza Cerca Terminale Aiuto
test@TestPart:~$ cat /etc/exports
# /etc/exports: the access control list for filesystems which may be exported
# to NFS clients. See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4 gss/krb5l(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes gss/krb5l(rw,sync,no_subtree_check)
#
/home/mpiuser/sharedDir *(rw,sync,no_root_squash,no_subtree_check)
test@TestPart:~$
```

Fatto questo possiamo riavviare il nostro server, digitiamo quindi:

**“sudo service nfs-kernel-server restart”**.

#### **4) Condivisione cartella NFS su SLAVE**

Dal nostro slave installiamo il pacchetto **common** digitando : **“ sudo apt-get install nfs-common ”** .

Creiamo la cartella con lo stesso nome della cartella condivisa : **“mkdir sharedDir”**.

Adesso dobbiamo montare la cartella condivisa, presente nel master :

**“sudo mount -t nfs master:/home/mpiuser/ sharedDir ~/ sharedDir”**.

Per controllare se la cartella risulta montata digitiamo : **“df -h”**, nella lista delle cartelle dovrebbe comparire l’indirizzo IP, o il nome associato all’indirizzo IP qualora voi abbiate seguito il passo 0, seguito dal percorso della directory condivisa.

Per rendere permanente la condivisione, in modo da non dover ripetere il passaggio 3.2 ad ogni riavvio, digitiamo :

**“sudo -i gedit /etc/ fstab”** e modifichiamo il file in modo da avere un risultato simile al seguente :

**#MPI CLUSTER SETUP**

**master:/home/mpiuser/sharedDir /home/mpiuser/sharedDir nfs**

Fatto questo abbiamo terminato con il setup.

#### **5) Come avviare i programmi MPI in modo da farli girare sul nostro Cluster?**

Per testare il corretto funzionamento del cluster possiamo accingerci a avviare un semplice programma, ad esempio un “Hello World” parallelizzato. Per accorgevi se la CPU dello slave è usata dal master possiamo utilizzare Htop, un task manager, maggiori info <https://hisham.hm/htop/> .

Per installare Htop digitare **“sudo apt-get install htop”** ed avviatelo.

Inserire il programma nella cartella condivisa e compilarlo direttamente nella stessa.

Per avviare il programma in parallelo digitiamo **“mpirun -np 2 -hosts master, slave ./a.out”**.

Dove **2** è il numero di processi. Master e slave posso essere sostituiti dai rispettivi IP.

In Htop vedremo i core lavorare e sulla console del master otterremo l’output del nostro programma.

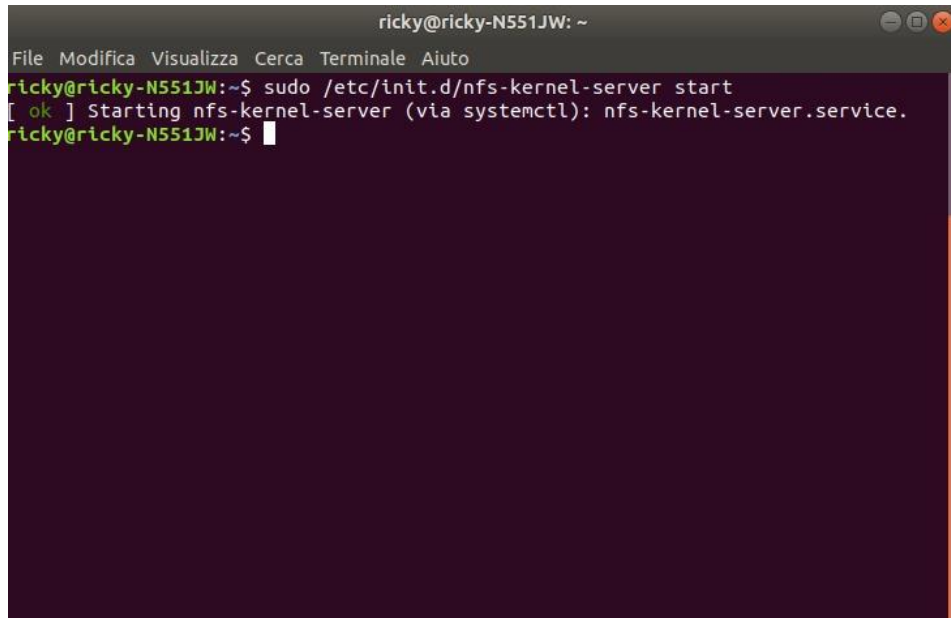
Nel caso nostro caso, HelloWorldParallelo con 2 processi, vedremo che un processo sarà gestito dal master e l’altro dallo slave.

Ripetiamo la lista di comandi per avviare un programma correttamente in cluster una volta configurato :

Dal master:

Dobbiamo avviare il server NFS dal nostro account principale, dunque :

start NFS “`sudo /etc/init.d/nfs-kernel-server start`”

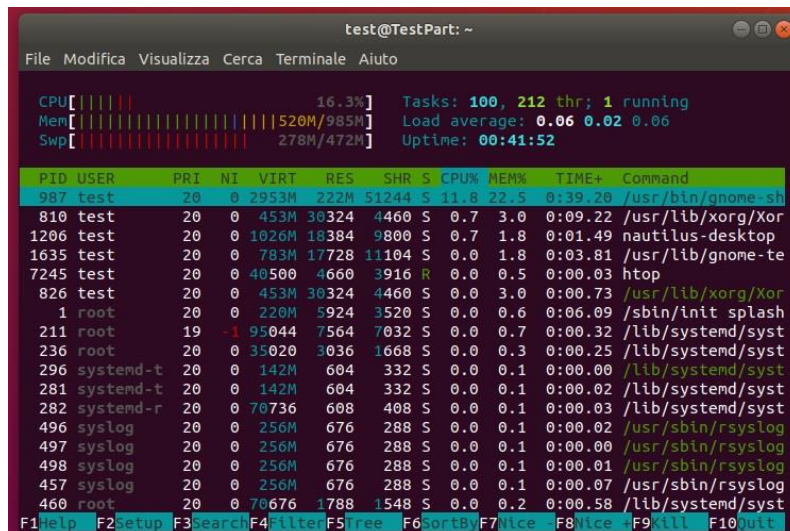


```
ricky@ricky-N551JW: ~  
File Modifica Visualizza Cerca Terminale Aiuto  
ricky@ricky-N551JW:~$ sudo /etc/init.d/nfs-kernel-server start  
[ ok ] Starting nfs-kernel-server (via systemctl): nfs-kernel-server.service.  
ricky@ricky-N551JW:~$
```

Dallo Slave:

Montiamo la cartella con : “`sudo mount -t nfs master:/home/mpiuser/ sharedDir ~/ sharedDir`” e  
verifichiamo con “`df -h`”.

Avviamo **htop** per visualizzare l’andamento della cpu.



```
test@TestPart: ~  
File Modifica Visualizza Cerca Terminale Aiuto  
  
CPU[|||||] 16.3% Tasks: 100, 212 thr; 1 running  
Mem[|||||] 520M/985M Load average: 0.06 0.02 0.06  
Swp[|||||] 278M/472M Uptime: 00:41:52  
  
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command  
987 test 20 0 2953M 222M 51244 S 11.8 22.5 0:39.20 /usr/bin/gnome-sh  
810 test 20 0 453M 30324 4460 S 0.7 3.0 0:09.22 /usr/lib/xorg/Xor  
1206 test 20 0 1026M 18384 9800 S 0.7 1.8 0:01.49 nautilus-desktop  
1635 test 20 0 783M 17728 11104 S 0.0 1.8 0:03.81 /usr/lib/gnome-te  
7245 test 20 0 40500 4660 3916 R 0.0 0.5 0:00.03 htop  
826 test 20 0 453M 30324 4460 S 0.0 3.0 0:00.73 /usr/lib/xorg/Xor  
1 root 20 0 220M 5924 3520 S 0.0 0.6 0:06.09 /sbin/init splash  
211 root 19 -1 95044 7564 7032 S 0.0 0.7 0:00.32 /lib/systemd/syst  
236 root 20 0 35020 3036 1668 S 0.0 0.3 0:00.25 /lib/systemd/syst  
296 systemd-t 20 0 142M 604 332 S 0.0 0.1 0:00.00 /lib/systemd/syst  
281 systemd-t 20 0 142M 604 332 S 0.0 0.1 0:00.02 /lib/systemd/syst  
282 systemd-r 20 0 70736 608 408 S 0.0 0.1 0:00.03 /lib/systemd/syst  
496 syslog 20 0 256M 676 288 S 0.0 0.1 0:00.02 /usr/sbin/rsyslog  
497 syslog 20 0 256M 676 288 S 0.0 0.1 0:00.00 /usr/sbin/rsyslog  
498 syslog 20 0 256M 676 288 S 0.0 0.1 0:00.01 /usr/sbin/rsyslog  
457 syslog 20 0 256M 676 288 S 0.0 0.1 0:00.07 /usr/sbin/rsyslog  
460 root 20 0 70676 1788 1548 S 0.0 0.2 0:00.58 /lib/systemd/syst
```

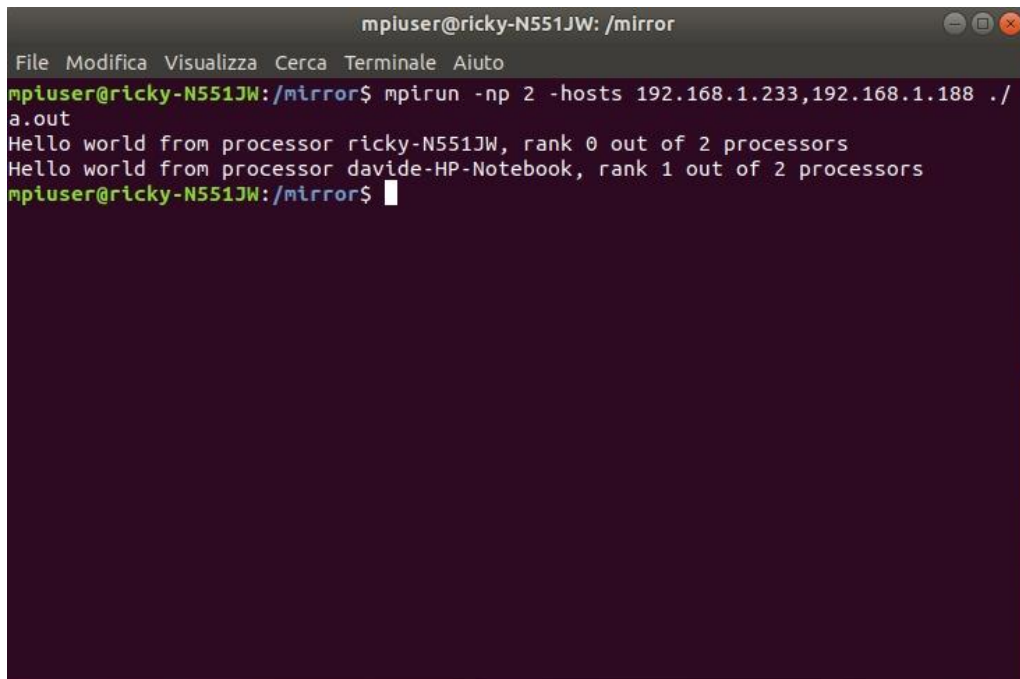
### Dal master:

Facciamo il login sul nostro account creato con “su mpiuser”.

**NB:** Ovviamente non dimentichiamo il lamboot su entrambi i PC.

Navighiamo nella cartella condivisa : “cd /sharedDir”.

Compiliamo il programma con “mpic++ helloMPI.cpp” ed avviamolo con “mpirun -np 2 -hosts master, slave ./a.out”.



```
mpiuser@ricky-N551JW: /mirror
File Modifica Visualizza Cerca Terminale Aiuto
mpiuser@ricky-N551JW:/mirror$ mpirun -np 2 -hosts 192.168.1.233,192.168.1.188 ./a.out
Hello world from processor ricky-N551JW, rank 0 out of 2 processors
Hello world from processor davide-HP-Notebook, rank 1 out of 2 processors
mpiuser@ricky-N551JW:/mirror$
```

### FAQ & TIPS

#### E' necessario che i processori abbiano la stessa architettura?

**NO**, in pratica avere la stessa architettura non è necessario ma POTREBBE non garantire la stessa scalabilità di un cluster con risorse “Gemelle”.

#### Come faccio a digitare la tilde per il passaggio 2?

Basta premere f5.

#### Posso compilare usando allegro?

**SI!** Basta digitare : “mpic++ example.cpp -lallegro -lallegro\_image”

La guida è stata creata sulla base di un altro tutorial che potrete trovare al seguente LINK:

<http://mpitutorial.com/tutorials/running-an-mpi-cluster-within-a-lan/>

Alcuni dei passaggi, in particolare quelli finali, non permettevano il corretto funzionamento del cluster. Solo gli innumerevoli TEST ci hanno consentito di riuscire nell'intento.

“La pazienza, la perseveranza e il sudato lavoro creano un'imbattibile combinazione per il successo.”