

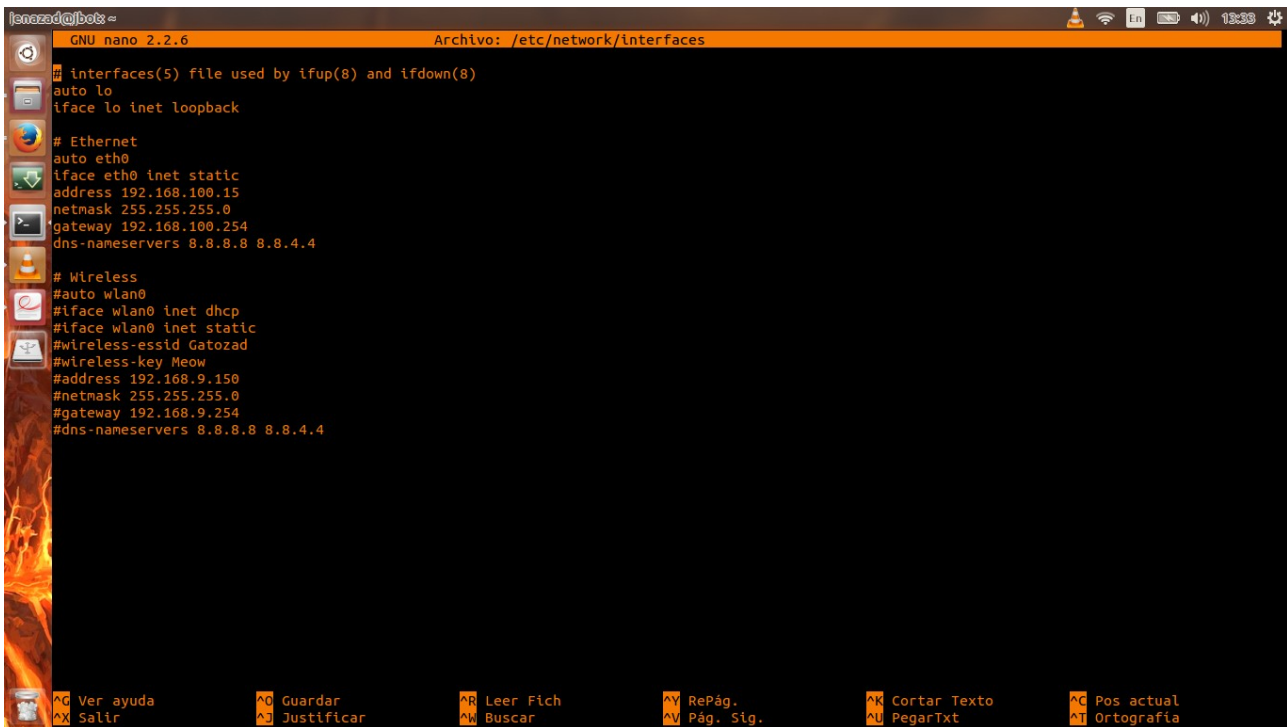
Construccion de un CPU Cluster tipo Beowulf

Nuestro sistema sobre el cual trabajaremos sera ubuntu.

Cuando hayan instrucciones con \$ delante, son instrucciones que se escribieran en shell.

Primero, configuramos nuestro IP, DNS, etc ...

\$sudo nano /etc/network/interfaces



```
GNU nano 2.2.6 Archivo: /etc/network/interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

# Ethernet
auto eth0
iface eth0 inet static
address 192.168.100.15
netmask 255.255.255.0
gateway 192.168.100.254
dns-nameservers 8.8.8.8 8.8.4.4

# Wireless
#auto wlan0
#iface wlan0 inet dhcp
#iface wlan0 inet static
#wireless-essid Gatozad
#wireless-key Meow
#address 192.168.9.150
#netmask 255.255.255.0
#gateway 192.168.9.254
#dns-nameservers 8.8.8.8 8.8.4.4

^G Ver ayuda  ^O Guardar    ^R Leer Fich  ^Y RePág.    ^K Cortar Texto ^C Pos actual
^X Salir      ^J Justificar ^B Buscar     ^P Pág. Sig. ^U PegarTxt    ^T Ortografia
```

Luego, debemos saber cuantas computadoras usaremos para con el cluster.

Supongamos que tenemos 1 computadora destinada a ser master(localhost) y 4 computadoras destinadas a ser slaves llamados ub0, ub1, ub2 y ub3.

Lo primero que debemos hacer es que en cada una de dichas computadoras se modifique el archivo /etc/hosts de la siguiente manera:

\$sudo nano /etc/hosts

donde debemos modificar de esta manera:

```
127.0.0.1    localhost
192.168.133.100 ub0
192.168.133.101 ub1
192.168.133.102 ub2
192.168.133.103 ub3
```

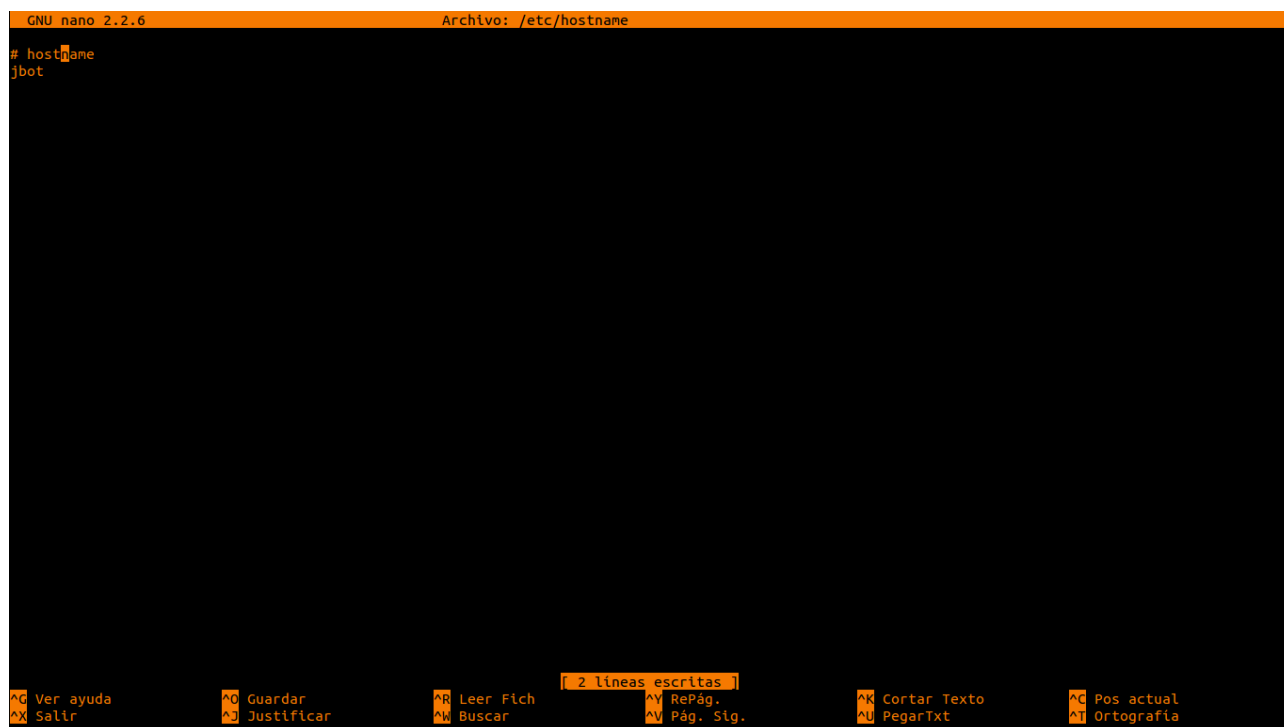
```
GNU nano 2.2.6                               Archivo: /etc/hosts                               Modificado
127.0.0.1    localhost
127.0.1.1    jbot
192.168.133.100 ub0
192.168.133.101 ub1
192.168.133.102 ub2
192.168.133.103 ub3

# The following lines are desirable for IPv6 capable hosts
::1          ip6-localhost ip6-loopback
fe00::0      ip6-localnet
ff00::0      ip6-mcastprefix
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters

^G Ver ayuda      ^O Guardar      ^R Leer Fich     ^V RePág.       ^K Cortar Texto  ^C Pos actual
^X Salir          ^D Justificar   ^W Buscar        ^V Pág. Sig.    ^U PegarTxt      ^T Ortografía
```

Y podemos modificar nuestro nombre de PC en /etc/hostname

\$sudo nano /etc/hostname



```
GNU nano 2.2.6 Archivo: /etc/hostname
# hostname
jbot

^G Ver ayuda      ^O Guardar      ^R Leer Fich     | 2 líneas escritas |  ^Y RePág.      ^K Cortar Texto  ^C Pos actual
^X Salir          ^J Justificar   ^B Buscar        ^V Pág. Sig.     ^U PegarTxt      ^T Ortografía
```

Intalacion NFS

Que es NFS?

NFS (Network File System) es un protocolo de sistema de archivos distribuidos que permite a un usuario en un cliente ordenador acceder a archivos a traves de una red tanto como se puede acceder al almacenamiento local.

NFS se basa en ONC RPC (Open Network Computing Remote Prodecidure Call).

Prodecemos a instalar en el master y todos los slaves:

\$sudo apt-get install nfs-server nfs-client

Ahora compartimos una carpeta desde el master hacia todos los slaves (cuyos Ips están en el archivo hosts), dela siguiente manera.

Creamos una carpeta en el master y todos los slaves:

\$sudo mkdir /forShare

Y luego compartir el contenido de esta carpeta situada en el nodo maestro a todos los otros nodos. Para hacerlo, primero edite el archivo / etc / exports en el nodo maestro para contener la línea adicional

```
$echo "/mirror *(rw,sync)" | sudo tee -a /etc/exports
```

reiniciamos el servidor nfs

```
$sudo service nfs-kernel-server restart
```

Montando carpetas del master a slaves:

ub0

```
$sudo mount localhost:/mirror /mirror
```

ub1

```
$sudo mount localhost:/mirror /mirror
```

ub2

```
$sudo mount localhost:/mirror /mirror
```

ub3

```
$sudo mount localhost:/mirror /mirror
```

Creamos un usuario para la carpeta mirror :

```
$sudo adduser mpiu /mirror
```

```
$sudo passwd mpiu
```

Instalacion SSH server

Que es SSH ?

Es un protocolo de red criptográfico para iniciar sesiones basado en shell en maquinas remotas de manera segura.

Esto permite al usuario ejecutar comandos en el símbolo del sistema de la máquina sin que ellos estén presentes físicamente cerca de la máquina.

También permite al usuario establecer un canal seguro sobre una red insegura en un cliente-servidor arquitectura, la conexión de un cliente SSH aplicación con un servidor SSH .

SSH se suele utilizar para iniciar sesión en una máquina remota y ejecutar comandos.

Que es SSH server ?

Es un programa de software que utiliza el protocolo SSH para aceptar conexiones desde equipos remotos.

SFTP / SCP transferencias de archivos y control remoto conexiones de los terminales son los casos de uso popular para un servidor SSH.

Instalamos SSH server:

```
$sudo apt-get install openssh-server
```

Es un protocolo de red criptográfico para iniciar sesiones basado en shell en maquinas remotas de manera segura.

Esto permite al usuario ejecutar comandos en el símbolo del sistema de la máquina sin que ellos estén presentes físicamente cerca de la máquina.

También permite al usuario establecer un canal seguro sobre una red insegura en un cliente-servidor arquitectura, la conexión de un cliente SSH aplicación con un servidor SSH .

SSH se suele utilizar para iniciar sesión en una máquina remota y ejecutar comandos sino que también apoya un túnel , reenvío de puertos TCP y X11 conexiones se pueden transferir archivos utilizando el asociado de transferencia de archivos SSH (SFTP) o copia segura protocolos (SCP).

SSH utiliza el modelo cliente-servidor.

En resumen, SSH es un protocolo utilizado para control remoto de otros equipos de la red y para transferencia cifrada de ficheros.

Public key: Encripta texto plano

Private key: Decrypta texto plano

SSH utiliza criptografía de llave publica para autenticar el equipo remoto y permitir que autorice al usuario entrante.

Hay 2 maneras:

1. la primera es que ambas partes usen llaves publicas-privadas generadas automáticamente y cifrar una conexión red y luego pedir contraseña de autenticación para iniciar sesión.

2. que se utiliza un par de llaves publicas generadas manualmente, permitiendo a usuario/programas iniciar sesión sin pedir llave.

La llave publica se coloca en todos los equipos que deben permitir acceso al titular de la llave privada.

Es decir, que la persona que ofrece la PublicKey también posee PrivateKey.

Luego de instalar SSH, se deberá crear una clave del ordenador correspondiente:

```
$sudo ssh-keygen -t rsa -f /etc/ssh/ssh_host_rsa_key
```

```
$sudo ssh-keygen -t dsa -f /etc/ssh/ssh_host_dsa_key
```

```
$sudo ssh-keygen -t ecdsa -f /etc/ssh/ssh_host_ecdsa_key
```

Configurando SSH:

Configuracion:

```
man sshd_config
```

Antes de tocar ese archivo haremos una copia de seguridad

```
$sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.original
```

```
$sudo chmod aw /etc/ssh/sshd_config.original
```

```
$sudo gedit /etc/ssh/sshd_config
```

-Tener sshd permita credenciales de acceso basados en clave pública, simplemente añadir o modificar la línea:

```
PubkeyAuthentication yes
```

-Añadiremos la siguiente línea para permitir el acceso mediante SSH al usuario que tengas:

```
AllowUsers meow
```

-Si queremos permitir el acceso mediante SSH al superusuario root añadiremos la siguiente línea a este fichero:

```
PermitRootLogin yes, Aunque esto no es muy recomendable por cuestiones de seguridad.
```

-Para indicar con qué interfaz de red se establecerá la conexión (en el caso de que tengamos varias interfaces de red con varias IPs en nuestro servidor de Linux Ubuntu) añadiremos la línea:

```
ListenAddress 192.168.1.254
```

-También podremos cambiar el puerto por defecto para SSH (es recomendable por seguridad), que es el 22, añadiendo (o modificando si existe) la línea:

```
Port 2222
```

-Ahora para hacer correctos los cambios :

```
$sudo /etc/init.d/ssh restart
```

```
$sudo /etc/init.d/ssh start
```

```
$sudo /etc/init.d/ssh stop
```

CREACION DE SSH-KEY

Están ubicadas en :

`$HOME /.ssh/identity`

aquí están las llaves privadas RSA cuando se usa el protocolo SSH versión 1

`$HOME /.ssh/identity.pub`

contiene la clave pública RSA para la autenticación cuando se utiliza la versión del protocolo SSH 1.

`$HOME /.ssh/id_dsa`

aquí están las llaves privadas DSA cuando se usa el protocolo SSH versión 2

`$HOME /.ssh/id_dsa.pub` contiene la clave pública DSA para la autenticación cuando se utiliza la versión del protocolo SSH 2.

`$HOME /.ssh/id_rsa`

aquí están las llaves privadas RSA cuando se usa el protocolo SSH versión 2

`$HOME /.ssh/id_rsa.pub`

contiene la clave pública RSA para la autenticación cuando se utiliza la versión del protocolo SSH 2.

Ahora damos permisos a las siguientes carpetas

para que puedan escribirse en procesos:

`$sudo chmod 700 ~/.ssh`

`$sudo chmod 600 ~/.ssh/authorized_keys`

Para generar la conexión hacia otra PC llamada PCexterna desde la nuestra llamada PCnuestra haremos:

En nuestra PC nos conectaremos a otras pc's con el comando ssh y la dirección IP a donde queremos ir.

```
$ssh PCexterna@ipexterno
```

pedirá contraseña de PCexterna para acceder, y listo

Para agregar a nuestra maquina(PCnuestra) llaves de otra maquina (PCexterna):

añade la llave publica de nuestro origen(PCnuestra) a las del servidor destino(PCexterna)

```
$cat ~/.ssh/id_rsa.pub | ssh PCexterna@IPexterna "cat>>.ssh/authorized_keys"
```

Y a la vez en la PCexterna:

```
$ssh PCnuestra@ipnuestra
```

pedirá contraseña de PCnuestra para acceder, y listo.

Para agregar a la maquina(PCexterna) llaves de nuestra maquina (PCnuestra):

añade la llave publica de nuestro origen(PCexterna) a las del servidor destino(PCnuestra):

```
$cat ~/.ssh/id_rsa.pub | ssh PCnuestra@IPnuestra "cat>>.ssh/authorized_keys"
```

Y listo, ahora entre ambas PC's se podran conectar sin necesidad de pedir contraseña.

(hacer esto SOLO con las PC's que quieras que accedan y a las cuales acceder, recuerda que si haces esto a un conjunto de PC's, todas podrán acceder a tu sistema sin filtro alguno).

Existe una manera de administrar todas las PC's de manera remota, sin necesidad de acceder a una por una, y esa manera se le atribuye a un software llamado clusterssh.

Ejemplo de una private Key: en /.ssh/ se almacenan id_rsa (privadas)

```
jenazad@jbot: ~/.ssh$ cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAw+ZIuksAAz9zGHEApGCpyEekDt1KWFUrYzzYMH04rJ7s+g76
/8Nl/j4n4wpddfkftfgNeEYqRm5vV9qUUNtkgpxUX0AZ+onK8VD648Zq79J2Bc67
3P6AUI+MXZ/3BmYz8/n7U8qgdJKIwLtf45GVNq0Yz9em34rcyAzDsApGCwvKY+g
jre0bGxtMQysW7GEu0CeDpD0E56BeFL9ovj3yJLwuy90h5y8qLD7m07EyAlMYanq
OPKY0n3Som4K6hLtGczBPPT06F8gMfKdnFokFHocCUBLzp/hfHLUDD9bjm6LdIND
50tEzV3/Jyfy99qDPkhTwLUXJdhQj5Cn04B3aQIDAQABAoIBADx/Tn8xnvRh0Np4
c1LQHL6KGS8Z+c6jpr2D2cPGyt/MSKTEoZEatv6LFwHRGCHJrNngkllaeJp8gnsa
HnmF91k6l/c0qh6yFQFHH+Pdr8KWPSP3+J473kaCvjdf0jqCrLVKlHZJ33euCIuf
cRg1LgLGZ96QgFgGs8Ze6EWL5E35NDWxJYMEFE/OMFzSYeIVFrXJntvYx4b4k60D
SEbj2FEWjJRI0tC+dUckhpDyZXvR9TM9MyPbfDAEXZgwK90gbamQ1cf/wvrV/4l
AMa7TawvpzJvK+tvVW+mW3zACKsK1m98GLJF2tzF88qP/fvM50kkvo3umxlgZNYI
bqcKhLEcQYEA9h9UCFBk9P5bGQqK75a6+hEcDM4Gg4gEwhszJEj8TySA9UzXZ1R8
xR0tqFTB02B8tuhN/AUHYJvEoPKm50m26cT5raALd7gRCVgac0qWlayJUBp8cSB
RxM0Q0gqbAFz/qg3eHXSKTtpUcl9xLn+d25H0JeJN1wqW7n8JjMpkUCgVEAY8L2
4JSEHe+BqThYTORgqrSpmYmfS9B0SIVg7oKbkW65dqMVF2oe8LONdxJ5h0QAiUt
AlF/z1aJh1zz0Q5qJF4/LmSYNXf7FILuYDWUK9ZdLXbhemzXGCELWH/6SLZZczr
+lnWFbN8P6eJG/V57cVGHthtFq+6IH6bRLSD0NUCgYA94Fag+IXJy8Jf83Sm2Bwf
D/o7hFnb8TIVvuJ0Lwpwd0hH5m3V1d4vp4KA2jhgDa7kvzAz/LswaEFBKIRrkQa6J
7IrS1veguFkgSbbKIPZ0IHieQSn1LgRLOI45uqxFS6Wgm+8GjKLA0L1UnWrv8mc6
PfnzPb9qCXtwpXj0N70M+QKBGCV7NHmLfLpnP6uH5HJnS725PL67TE0jC9jooM56
WbeSAwN3VWYv0Lpqy0g4vkendpHLwh7NkFjMqLQscPs6fAzAHJsp2LF0jqw+wfv
p1wt5zo9ew7+MxVRX9Vbv8t4hI2vAdgt5fJHfBoF1qmWGKPTHJLAMSrbcJe+dpb4
mJVBaoGBALKE/B7jo6u0/JvVoF003x0XUNBP7a5yL+HjIi8Za9Wpxr2c8edkkt75
rWwORnqj0EQFE4WhFR1yanjwb5S37j7d5QWj9j7M00FFQ5X+S6btUJ/i114DetY/
2tNHXYhprqamSYj0dBX3C4deIsRod104DJQNMgx6+yL4ZPvuR+RK
-----END RSA PRIVATE KEY-----
jenazad@jbot: ~/.ssh$
```

Ejemplo de una public key en /.ssh se almacenan en id_rsa.pub

```
jenazad@jbot: ~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDD5k16SwADP3MYcQCKyKnIR6Q03UpYVstjPNgweJtsnuz6Dvr/w2L+PlfjCL11+S1+A14Rlqszm9VX2pRQ225CnFRc4DP6tcrxUPrjxmrv0nYFzrv
c/oBQj4xdn/cFaZhnz+ftTyqB8kojCw0XhIZU2o5jP16bfltZIDM0wCkYLC8pj6Cot7RsZe0xDKXbsY57QJ4M90MTnoF4WX2L+PFKMvC7L06FLLyqUPubTsTICUxhqe048pJ5fdKlBgrqEu0ZzME890
joXyAx8p2cWlQ0ehWJQEv0n+F8evQMP1u0bot0gNLS0TNXF8nJ9j2uoM+SFPCJRCrL2FCpkKfTgHdp jenazad@jbot
jenazad@jbot: ~/.ssh$
```

Instalando CSSH

En consola escribir

```
$sudo apt-get install clusterssh
```

configurando cssh:

Es una aplicación para la administración de ordenadores conectados en red permitiendo interactuar con ellos a través de SSH con dos o mas ordenadores simultáneamente desde la misma herramienta.

Crear un archivo en la carpeta ~/etc/

llamado clusters y dentro de ese archivo escribir

Por ejemplo, digamos que tengo dos grupos, cada uno compuesto por dos máquinas.

"Clúster1" tiene las máquinas "Test1" y "Test2" en ella, y "Cluster2" tiene las máquinas "Test3" y "Test4" en ella.

El ~ .csshrc (o / etc/clusters) archivo de control se vería así:

```
clusters = cluster1 cluster2
```

```
cluster1 = test1 test2
```

```
cluster2 = test3 test4
```

Donde cluster1 y cluster 2 son los nombres de los clusters con las maquina test1, ... si quisiéramos uno que contenga todo:

```
clusters = all
```

```
all = test1 test2 test3 test4
```

o mejor aun

```
clusters = cluster1 cluster2 all
```

```
cluster1 = test1 test2
```

```
cluster2 = test3 test4
```

```
all = cluster1 cluster2
```

y ejecutamos en consola

```
cssh -l <username> <clustername> &
```

El parámetro "&" es para cortar la salida o también sin el archivo clusters

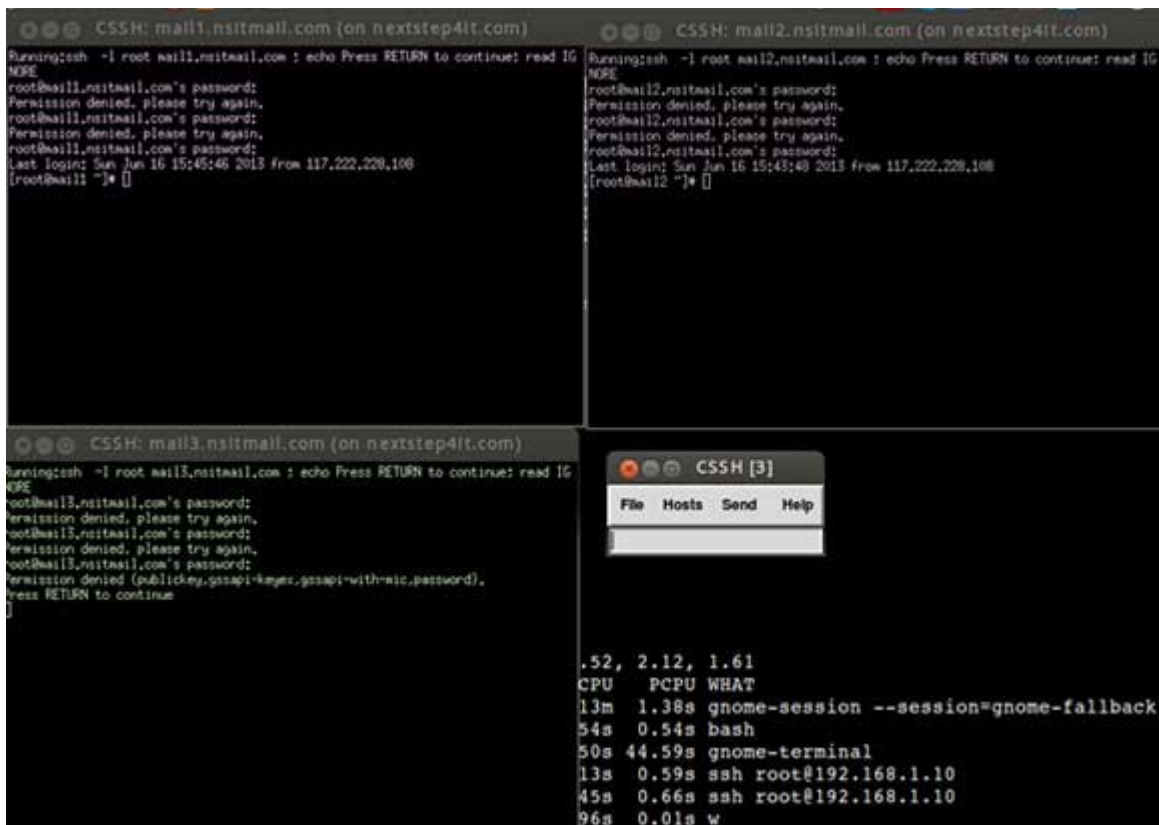
```
cssh -l <username> <machine1> <machine2> <machine3> ... &
```

o tambien de manera directa (con el archivo /etc/clusters)

```
cssh <clustername>
```

Aparecerá una ventana blanca pequeña y un número de ventanas(shell) según el número de máquinas en tu cluster, y notaremos que escribiendo en esa, lo que escribamos se copia en las demás consolas, entonces instalamos en todas a la vez).

Ejemplo de cssh:



```
Running:cssh -l root mail1,nsitmail.com : echo Press RETURN to continue: read IG NORE
root@mail1,nsitmail.com's password:
Permission denied, please try again.
root@mail1,nsitmail.com's password:
Permission denied, please try again.
root@mail1,nsitmail.com's password:
Last login: Sun Jun 16 15:45:46 2013 from 117.222.228.108
[root@mail1 ~]#
```

```
Running:cssh -l root mail2,nsitmail.com : echo Press RETURN to continue: read IG NORE
root@mail2,nsitmail.com's password:
Permission denied, please try again.
root@mail2,nsitmail.com's password:
Permission denied, please try again.
root@mail2,nsitmail.com's password:
Permission denied, please try again.
root@mail2,nsitmail.com's password:
Last login: Sun Jun 16 15:45:48 2013 from 117.222.228.108
[root@mail2 ~]#
```

```
Running:cssh -l root mail3,nsitmail.com : echo Press RETURN to continue: read IG NORE
root@mail3,nsitmail.com's password:
Permission denied, please try again.
root@mail3,nsitmail.com's password:
Permission denied, please try again.
root@mail3,nsitmail.com's password:
Permission denied, please try again.
root@mail3,nsitmail.com's password:
Permission denied (publickey,gssapi-keyex,gssapi-withmic,password).
Press RETURN to continue.

```

```
File Hosts Send Help

.52, 2.12, 1.61
CPU PCPU WHAT
13m 1.38s gnome-session --session=gnome-fallback
54s 0.54s bash
50s 44.59s gnome-terminal
13s 0.59s ssh root@192.168.1.10
45s 0.66s ssh root@192.168.1.10
96s 0.01s w
```

Instalacion de MPICH

Ahora usaremos nuestro cluster para un cálculo sencillo, pero primero instalamos en consola escribir:

```
$sudo apt-get install libcr-dev mpich2
```

Lo hacemos en todas las maquina, o mejor aun, usando cssh para acelerar el proceso.

O también, montamos la carpeta desde el maestro hacia los nodos de la siguiente manera:

En el nodo escribir:

```
$sudo mount master:/... /...
```

donde: /... es la dirección donde se encuentra MPICH.

MPI: Message Pasing Interface

Es un protocolo de comunicaciones independiente del lenguaje usado para programar computadoras paralelas.

MPI es una interface Sistema Distribuido de Paso de Mensajes

, Es un estándar de paso de mensajes facto para la comunicación entre procesos que modelan un programa paralelo que se ejecuta en un sistema de MEMORIA DISTRIBUIDA.

También modos sincronizados y asincronizados de comunicación punto a punto y comunicación colectiva.

Comunicación paso mensajes entre procesadores o nodos.

Para compilar un programa usamos:

```
mpicc nombre_archivo.c -o execute
```

La manera de compilación es similar a como lo hace GCC (GNU Collection Compiler).

Para ejecutar el binario:

```
$mpiexec -n 1 ./execute
```

Solo ejecutará en nuestra máquina, pero si quisiéramos mandar los datos a otras máquinas, lo hacemos de la siguiente manera:

Para mandar mensajes a otras maquina usar

```
$mpiexec -f machinefile -n 4 ./a.out
```

Donde machine file es el archivo donde se almacenan los ips de las demás maquinas., como se muestra a continuacion

```
#ub0  
192.168.133.100  
#ub1  
192.168.133.101  
#ub2  
192.168.133.102  
#ub3  
192.168.133.103
```