Installation d'un cluster HPC

Jean François QUINET

21 février 2020

Après de nombreuses recherches infructueuses sur le net pour construire un cluster de calcul, j'ai décidé d'improviser et d'en faire profiter ceux qui seraient intéressés. Un des objectifs était d'utiliser au maximum des matériels de récupération, le coût total de ce cluster est inférieur à 100 €. L'architecture logicielle basée sur Linux nécessite une bonne connaissance de ce système et des réseaux, mais un débutant peut très bien s'en sortir en fouillant un peu dans la documentation disponible sur le net.
Ce cluster est destiné au calcul de prévision météo avec l'application WRF-ARW, mais peut-être utilisé avec toute autre application

I) installation d'IPXE

Pxe est plus simple à mettre en oeuvre mais n'est pas très stable pour le démarrage de plusieurs noeuds sur un SAN, j'ai donc choisi de mettre en oeuvre IPXE

1) Installation de DHCP server

Installer isc-dhcp-server
apt-get install isc-dhcp-server
configurer le fichier /etc/dhcp/dhcpd.conf
spécifier les paramètres généraux
configurer les options PXE, se référer à la documentation ex: http://doc.ubuntu-fr.org/ipxe
configurer le réseau conformément à son architecture réseau. Attention au paramètre à ne pas manquer next-server <ip address>
enfin indiquer les programmes qui seront exécutés en fonction de l'architecture réseau au moment du boot réseau

```
root@cluster1:~# cat /etc/dhcp/dhcpd.conf
# option definitions common to all supported networks...
option domain-name "xxx.fr";
option domain-name-servers 192.168.xxx.xxx;

default-lease-time 86400;
max-lease-time 604800;
###PXE###
option space PXE;
option PXE.mtftp-ip code 1 = ip-address;
option PXE.mtftp-cport code 2 = unsigned integer 16;
option PXE.mtftp-sport code 3 = unsigned integer 16;##
...... (voir tous les paramètres dans la documentation)

# RESEAUX

subnet 192.168.xxx.0 netmask 255.255.255.0 {
        option broadcast-address 192.168.xxx.255;
        option domain-name-servers 192.168.xxx.xxx;
        range 192.168.xxx.20 192.168.xxx.100;
        option routers 192.168.xxx.xxx;
        next-server 192.168.xxx.xxx;

# Adresse fixe

host cluster2 {
        hardware ethernet 01:02:03:04:05:06;
        fixed-address 192.168.xxx.xxx;
}
# PXE
        if option arch = 00:07 or option arch = 00:09 {
                if exists user-class and option user-class = "iPXE" {
                        filename "http://192.168.xxx.xxx/install.ipxe";
                } else {
                        filename "ipxe/ipxe.efi";
                }
        } else if option arch = 00:06 {
                if exists user-class and option user-class = "iPXE" {
                        filename "http://192.168.xxx.xxx/install.ipxe";
                } else {
                        filename "ipxe/ipxe32.efi";
                }
        } else {
                if exists user-class and option user-class = "iPXE" {
                        filename "http://192.168.xxx.xxx/install.ipxe";
                } else {
                        filename "undionly.kpxe";
                }
        }
}
```

Redémarrer le service dhcp

service isc-dhcp-server restart

2) Installation du serveur TFTP

apt-get install tftp-hpda

Adapter le fichier de configuration /etc/default/tftpd-hpa

```
# /etc/default/tftpd-hpa

TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/var/lib/tftpboot"
TFTP_ADDRESS="0.0.0.0:69"
TFTP_OPTIONS="--secure"
RUN_DAEMON="yes"
```

Relancer le serveur tftp
service tftpd-hpa restart

3) Installation du serveur Apache

apt-get install apach2 php libapache2-mod-php php-mysql php-curl php-gd php-itl php-json php-mbstring php-xml php-zip

Créer un lien symbolique du répertoire TFTP pour le serveur Apache

ln -s /var/lib/tftpboot /var/www/html/tftpboot

4) Installer les programmes de démarrage

Afin de simplifier l'interface entre les composants on va compiler les programmes afin d'y insérer le script chain.ipxe

Télécharger les sources nécessaires à la compilation

cd /tmp
git clone git://git.ipxe.org/ipxe.git

Créer le script chain.ipxe

nano /tmp/ipxe/src/chain.ipxe

```
# /tmp/ipxe/src/chain.ipxe

dhcp
chain http://192.168.xxx.xxx/install.ipxe
```

Installer si ce n'est déjà fait les paquets nécessaires à la compilation

apt-get install gcc binutils liblzma-dev make

Compiler les programmes:

cd /tmp/ipxe/src
make bin-x86_64-efi/ipxe.efi EMBED=chain.ipxe
make bin-i386-efi/ipxe.efi EMBED=chain.ipxe

make bin/undionly.kpxe EMBED=chain.ipxe

cp bin-x86_64-efi/ipxe.efi /var/lib/tftpboot/
cp undionly.kpxe /var/lib/tftpboot
cp bin-i386-efi/ipxe.efi /var/lib/tftpboot/ipxe32.efi

5) Création du script de démarrage

nano /var/www/html/install.ipxe

```
#!ipxe

set menu-timeout 9000000
set submenu-timeout ${menu-timeout}
isset ${menu-default} || set menu-default item3
set server_ip 192.168.xxx.xxx

menu
item --gap --            -------------SÉPARATION 1----------------
item item1               Demarrage ubuntu
item --gap --            -------------SÉPARATION 2----------------
item item2               Item 2
item item3               Item 3
item shell               Shell iPXE
item exit                Exit

choose --timeout ${menu-timeout} --default ${menu-default} target &&
goto ${target}

:item1
#Paramètres de démarrage pour item 1

:item2
#Paramètres de démarrage pour item 2

:item3
#Paramètres de démarrage pour item 3

:shell
shell

:exit
exit
```

La documentation concernant le langage de script est relativement succinct je vous invite néanmoins à la consulter sur http://ipxe.org

On complétera ce menu plus tard pour l'instant il permet de voir si tout fonctionne correctement jusque là. Pour s'en rendre compte il suffit de lancer une machine capable de booter sur le réseau.

II) installation ISCSI

Chaque noeud aura besoin de son propre espace disque. Pour se faire on installe un SAN sur le serveur PXE basé sur le protocol IP iscsi.

Soit on partage un espace du disque du serveur si on en possède qu'un soit on utilise un disque dédié à iscsi. (prévoir 10 GO par noeud)
On appelle la partie serveur ISCSI Target, et la partie client ISCSI Initiator.

1) Sur le serveur

apt-get update
apt-get install tgt lvm2

après l'installation des packages en session terminal user: root ou sudo
lsbkl -> affichage des disques

```
root@cluster1:/# lsblk
NAME                MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
loop0                 7:0    0  86,9M  1 loop /snap/core/4917
loop1                 7:1    0  89,1M  1 loop /snap/core/8039
sda                   8:0    0 465,8G  0 disk
├─sda1                8:1    0     1M  0 part
└─sda2                8:2    0   200G  0 part /
sr0                  11:0    1  1024M  0 rom
```

Inclure le disque affecté au SAN dans un LVM
pvcreate /dev/sdb
Créer un volume group: cluster étant le nom de mon volume
vgcreate cluster /dev/sdb
On peut vérifier le résultat par la commande
vgs

```
root@cluster1:/# vgs
  VG      #PV #LV #SN Attr   VSize    VFree
  cluster   1   1   0 wz--n- <149,05g <124,05g
```

Créer un volume logique
lvcreate -L <taille> (10GB) -n nom_volume_logique (clusterd1) nom_volume_group (cluster)
la taille du volume peut être spécifiée en % de l'espace disponible avec l'argument -l au lieu de -L
lvs pour vérification

```
root@cluster1:/# lvs
  LV        VG      Attr       LSize  Pool Origin Data%  Meta%  Move Log
Cpy%Sync Convert
  clusterd1 cluster -wi-ao---- 10,00g
```

Créer un LUN (Logical Unit Number) qui sera l'entité connectée par le client
Pour se faire créer un fichier de configuration sur le chemin /etc/tgt/conf.d
nano /etc/tgt/conf.d/cluster_iscsi.conf

```
<target iqn.xxxx.xxxxx.xx:lun1>
        # Provided device as an iSCSI target
        # cluster-clusterd1 correspond au Volume Group cluster
        # et Logical volume clusterd1
        backing-store /dev/mapper/cluster-clusterd1
</target>
```

D'autres paramètres optionnels peuvent être spécifiés en particulier pour la mise en oeuvre d'une authentification. (Pas nécessaire dans mon cas de figure toute l'installation étant sur un réseau dédié)

Redémarrer le service tgt
service tgt restart

On peut vérifier la configuration:
tgtadm --mode target --op show

```
root@cluster1:~# tgtadm --mode target --op show
Target 1: iqn.xxxx.xxxxx.xx:lun1
    System information:
        Driver: iscsi
        State: ready
    I_T nexus information:
    LUN information:
        LUN: 0
            Type: controller
            SCSI ID: IET     00010000
            SCSI SN: beaf10
            Size: 0 MB, Block size: 1
            Online: Yes
            Removable media: No
            Prevent removal: No
            Readonly: No
            SWP: No
            Thin-provisioning: No
            Backing store type: null
            Backing store path: None
            Backing store flags:
        LUN: 1
            Type: disk
            SCSI ID: IET     00010001
            SCSI SN: beaf11
            Size: 26844 MB, Block size: 512
            Online: Yes
            Removable media: No
            Prevent removal: No
```

la configuration est maintenant terminée.

2) Sur le client

On peut voir du coté client comment accéder au disque logique ainsi créé.

Pour commencer et si ce n'est pas déjà fait on installe open-iscsi, soit sur une machine distante soit sur la machine qui héberge iscsi-target
apt-get install open-iscsi

On peut alors visualiser la configuration du SAN
iscsiadm -m discovery -t st -p 192.168.xxx.xxx

```
root@cluster1:~# iscsiadm -m discovery -t st -p 192.168.xxx.xxx
192.168.xxx.xxx:3260,1 iqn.xxxx.xxxxx.xx:lun1
```

Cette commande crée un fichier (chemin /etc/iscsi/nodes) qui contiendra entre autres les informations d'authentification si présentes.

Pour se connecter sur le LUN on utilise toujours la commande iscsiadm

iscsiadm -m node -T 'iqn.xxxx.xxxxx.xx:lun1' - p 192.168.xxx.xxx -l

```
root@cluster1:/# iscsiadm -m node -T 'iqn.xxxx.xxxxx.xx:lun1' -p
192.168.xxx.xxx -l
Logging in to [iface: default, target: iqn.xxxx.xxxx.xx:lun1, portal:
192.168.xxx.xxx,3260] (multiple)
Login to [iface: default, target: iqn.xxxx.xxxxx.xx:lun1, portal:
192.168.xxx.xxx,3260] successful.
```

L'affichage des disques nous donne le résultat suivant

lsblk

```
root@cluster1:/# lsblk
NAME                MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
loop0                 7:0    0  89,1M  1 loop /snap/core/8039
loop1                 7:1    0  86,9M  1 loop /snap/core/4917
sda                   8:0    0 465,8G  0 disk
├─sda1                8:1    0     1M  0 part
└─sda2                8:2    0   200G  0 part /
sdb                   8:16   0 149,1G  0 disk
└─cluster-clusterd1 253:0    0    10G  0 lvm
sdd                   8:48   0    10G  0 disk
sr0                  11:0    1  1024M  0 rom
```

Notre disque de 10G est bien disponible, et après l'avoir partitionné on peut le monter sur un répertoire existant

```
root@cluster1:/# lsblk
NAME                 MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
loop0                  7:0    0  89,1M  1 loop /snap/core/8039
loop1                  7:1    0  86,9M  1 loop /snap/core/4917
sda                    8:0    0 465,8G  0 disk
├─sda1                 8:1    0     1M  0 part
└─sda2                 8:2    0   200G  0 part /
sdb                    8:16   0 149,1G  0 disk
└─cluster-clusterd1 253:0    0    10G  0 lvm
sdd                    8:48   0    10G  0 disk
└─sdd1                 8:49   0    10G  0 part
sr0                   11:0    1  1024M  0 rom
```

mount /dev/sdd1 /mnt/chroot

```
root@cluster1:/# ls -l /mnt/chroot
total 1214996
drwxr-xr-x   2 root root       4096 nov.  24 15:28 bin
drwxr-xr-x   3 root root       4096 nov.  24 16:08 boot
drwxr-xr-x   4 root root       4096 nov.  24 14:38 dev
drwxr-xr-x 134 root root      12288 nov.  25 17:54 etc
drwxr-xr-x   3 root root       4096 nov.  24 16:07 home
lrwxrwxrwx   1 root root         33 nov.  24 14:47 initrd.img ->
boot/initrd.img-4.15.0-70-generic
.......
```

Pour déconnecter le disque après l'avoir démonté
iscsiadm -m node -T 'iqn.xxxx.xxxxx.xx:lun1' - p 192.168.xxx.xxx -u

```
root@cluster1:/# iscsiadm -m node -T 'iqn.xxxx.xxxxx.xx:lun1' -p
192.168.xxx.xxx -u
Logging out of session [sid: 2, target: iqn.xxxx.xxxxx.xx:lun1,
portal: 192.168.xxx.xxx,3260]
Logout of [sid: 2, target: iqn.xxxx.xxxxx.xx:lun1, portal:
192.168.xxx.xxx,3260] successful.
```

Voila qui termine la partie configuration iscsi, pour plus d'info voir le site:
https://www.tecmint.com/setup-iscsi-target-and-initiator-on-debian-9/

III) Installation linux Ubuntu 18.04 sur le premier noeud

1) Créer la source d'installation

Télécharger l'image iso de la distribution à installer. Dans mon cas ubuntu 18.04, sur le site prendre Alternative download, puis Alternative ubuntu server installer, l' image standard ne dispose pas de toutes les options.

Si on dispose d'un lecteur CD on peut le faire directement en gravant l'image iso

Si ce n'est pas le cas on peut le faire avec un partage nfs et via le serveur http

Créer un répertoire ubuntu sur /var/lib/tftpboot
Décompresser l'image iso
Créer un répertoire temporaire pour extraire l'image iso (ex: /mnt/iso)
mount -o loop /<telechargement>/ubuntu-18.04.3-server.amd64.iso /mnt/iso
cp -av /mnt/iso/* /var/lib/tftpboot/ubuntu
umount /mnt/iso
rm -R /mnt/iso

Accorder les droits sur tous les fichiers nécessaire à l'installation
chown -R root:root /var/lib/tftpboot
chmod -R 755 /var/lib/tftpboot

et sur le lien symbolique créé plus haut, chapitre I paragraphe 3

2) Partage nfs

Partager le répertoire /var/lib/tftpboot/ubuntu sur nfs en adaptant le fichier /etc/exports. Se reporter à la documentation nfs si le service n'est pas déjà installé.

```
# /etc/exports: the access control list for filesystems which may be
exported
#               to NFS clients.  See exports(5).
#/var/tftpboot/ubuntu 192.168.20.0/255.255.255.0(rw,async
,no_root_squash,no_subtree_check)
```

3) Adapter le script du menu install.ipxe

nano /var/www/html/install.ipxe :

```
#!ipxe

set menu-timeout 9000000
set submenu-timeout ${menu-timeout}
isset ${menu-default} || set menu-default item3
set server_ip 192.168.xxx.xxx

menu
item --gap --            -------------SÉPARATION 1----------------
item installubuntu               Installation ubuntu
item --gap --            -------------SÉPARATION 2----------------
item item2               Item 2
item item3               Item 3
item shell               Shell iPXE
item exit                Exit

choose --timeout ${menu-timeout} --default ${menu-default} target &&
goto ${target}

:installubuntu
kernel http://${server_ip}/tftpboot/ubuntu/install/vmlinuz
initrd http://${server_ip}/tftpboot/ubuntu/install/netboot/
ubuntu-installer/amd64/initrd.gz
imgargs vmlinuz initrd=initrd root=/dev/nfs boot=install netboot=nfs
nfsroot=${server_ip}/var/lib/tftpboot/ubuntu ip=dhcp --
boot

:item2
#Paramètres de démarrage pour item 2

:item3
#Paramètres de démarrage pour item 3

:shell
shell

:exit
exit
```

Il ne reste plus qu'à booter la machine à installer, et en principe tout devrait se dérouler comme si on faisait une installation standard.

En cas de problème avec l'installation vérifier que l'on utilise bien le bon initial ramdisk, initrd.gz, en particulier pour la gestion des disques iscsi

4) Installation ubuntu

Durant l'étape de configuration des disques en principe on obtient le message: aucun disque trouvé ... et propose de sélectionner un driver. Prendre la première option proposée: s'identifier sur des cibles SCSI puis renseigner l' initiator name, puis l'adresse IP du serveur NAS . Le LUN précédemment créé s'affiche, le sélectionner et procéder au partitionnement. Inutile de prévoir une partition swap à partir de ubuntu 18.04. Si la détection du disque iscsi ne fonctionne pas ouvrir une session shell et se connecter sur le LUN avec la commande iscsistart
iscsistart -i <initiator name> - t < target name> -g <Target group (mettre 1) -a <adresse IP>

L'installation devrait se poursuivre normalement en créant un système sur le disque isci.
A la fin de l'installation choisir l'amorçage Grub
Ne pas rebooter le système, il faut auparavant modifier notre script install.ipxe afin de démarrer le système nouvellement installé sur le disque scsi.

5) Modification du script de boot

On peu réutiliser le menu créé précédemment et mettre par exemple en option 2 le démarrage de l'OS
Personnellement j'ai recréé un script qui n'affiche pas de menu et démarre sans halte l'OS

```
#!ipxe

set server_ip 192.168.20.160

# Demarrage cluster2
sanboot iscsi:192.168.xxx.xxx:::1:iqn.xxxxx.xxx.xx:lun1
```

Un paramètre important de la commande sanboot est l'URI qui se décompose ainsi
<servername>:<protocol>:<port>:<LUN>:<targetname>
<LUN> est le numéro en hexa de LUN que l'on peut afficher avec la commande utilisée précédemment :
tgtadm --mode target --op show

```
root@cluster1:~# tgtadm --mode target --op show
Target 1: iqn.xxxx.xxxxx.xx:lun1
    System information:
        Driver: iscsi
        State: ready
    I_T nexus information:
    LUN information:
        LUN: 0
            Type: controller
            SCSI ID: IET     00010000
.....................
        LUN: 1
            Type: disk
            SCSI ID: IET     00010001
            SCSI SN: beaf11
            Size: 26844 MB, Block size: 512
            Online: Yes
.....................
```

Ce numéro est différent en fonction des SAN utilisés il est donc important de bien identifier celui qui correspond à son installation

On peut maintenant démarrer notre nouveau système et l'adapter en fonction de ses besoins.

IV) Installation MPICH

L'exécution d'un programme sur plusieurs ordinateurs nécessite l'utilisation du protocole MPI (message passing interface) . Donc on installe MPICH qui est entre autre le plus répandu et celui qui est supporté par WRF.

Sur ubuntu:
apt-get install mpich

L'utilisation de mpich requiert un minimum de configuration:

1) SSH

Si ce n'est pas déjà fait il faut installer ssh sur toutes les machines
apt-get install openssh-server
En principe le client est déjà installé, si ce n'est pas le cas
apt-get install openssh-client

Pour se connecter sur une machine distante sans avoir à donner un mot de passe il faut installer des clés de sécurité.

Sur le client on génère une clé RSA
ssh-keygen -t rsa
On vérifie sur le répertoire ~/.ssh que l'on a bien entre autre 2 fichiers id_rsa: clé privée et id_rsa.pub: clé publique.

On envoie la clé publique sur le serveur (machine distante sur laquelle on veut se connecter)
ssh-copy-id <utilisateur distant>@<server>
<server> soit le nom du serveur, soit l'adresse IP
On vérifie sur le serveur que la clé a bien été implantée, dans le répertoire ~/.ssh de l'utilisateur distant on doit avoir un fichier authorized_keys

Sur le client on peut maintenant se connecter par ssh <utilisateur distant>@<server>
ou exécuter une commande sur le serveur distant par ssh <utilisateur distant>@<server> ls -l

2) NFS

En principe on a vu plus haut l'utilisation de NFS pour l'installation d' ubuntu.
On créé un répertoire sur la machine maître (serveur à partir duquel on va lancer le programme sur l'ensemble des noeuds) qui sera partagé avec tous les esclaves (serveurs qui vont participer à l'exécution du programme en parallèle)
Pour les options de partage dans le fichier /etc/exports on utilise (rw,sync,no_root_squash,no_subtree_check)

Installer nfs-common sur tous les esclaves
apt install nfs-common
Monter le répertoire sur tous les esclaves
mount -t nfs <serveur maître>:/<repertoire partagé> /<repertoire esclave>

3) Configuration hosts

Editer les fichiers /etc/fstab de chaque esclave pour rendre le mount permanent
Editer les fichiers /etc/hosts avec tous les noms des serveurs maîtres et esclaves

Créer un fichier de distribution des process dans le répertoire partagé ex
nano /<repertoire partagé>/machinefile
cluster0:4 # 4 process sur la machine cluster0
cluster1:2 # 2 process sur la machine cluster1
etc.....

4) Test de MPI

On peut tester notre installation en exécutant un petit programme. Récupérer le source du programme mpi_sample.c (on le trouvera sur http:// https://gist.github.com/varun-nagaraja/1227316)
Sur le serveur maître compiler le programme sur le répertoire partagé:
mpicc -o mpi_sample mpi_sample.c

Faire un premier test pour voir si tout fonctionne d'abord sur notre serveur maître
# mpiexec -n 2 -f machinefile /<repertoire partagé>/mpi_sample
n = nombre de process

```
root@linux:~# mpiexec -n 2 -f machinefile /mnt/partagenfs/mpi_sample
Hello MPI World From process 0: Num processes: 2
Hello MPI World from process 1!
```

Puis sur le serveur maître et les esclaves
mpiexec -n 5 -f machinefile /<repertoire partagé>/mpi_sample

```
root@linux:~# mpirun -np 5 /mnt/partagenfs/mpi_sample -hosts
192.168.xxx.xxx,192.168.xxx.xxx
Hello MPI World From process 0: Num processes: 5
Hello MPI World from process 1!
Hello MPI World from process 2!
Hello MPI World from process 3!
Hello MPI World from process 4!
```

Ca fonctionne on a exécuté le programme sur 5 process

V) Dupliquer les environnements

L'objectif ici est de créer un environnement par noeuds disponibles.

1) Créer un nouveau volume logique iscsi

commande **lvcreate** (cf chapitre II)

Ensuite créer un LUN en éditant le fichier

nano /etc/tgt/conf.d/cluster_iscsi.conf

Redémarrer le service tgt
service tgt restart
Vérifier la configuration:
tgtadm --mode target --op show

2) Installer le nouvel environnement

Booter le système avec le script du chapitre III paragraphe 3

Durand la configuration disque de l'installation (chapitre III paragraphe 4) sélectionner le LUN nouvellement créer

3) Modification du script install.ipxe
(chapitre III paragraphe 5)

nano /var/www/html/install.ipxe

```
#!ipxe

set server_ip 192.168.20.160

# Demarrage cluster2
iseq ${net0/mac} aa:aa:aa:aa:aa:aa && sanboot
iscsi:192.168.xxx.xxx:::1:iqn.xxxxx.xxx.xx:lun1 ||
# Demarrage cluster3
iseq ${net0/mac} bb:bb:bb:bb:bb:bb && sleep 30 && sanboot
iscsi:192.168.xxx.xxx:::1:iqn.xxxxx.xxx.xx:lun2 ||

exit
```

Le script s'est enrichi de quelques nouvelles commandes:
iseq qui permet de tester une égalité, ici en fonction de l'adresse mac de l'adaptateur réseau on lance la commande sanboot
Les commandes sont chaînées avec l'opérateur &&. L'opérateur || est obligatoire même si on a pas besoin d'exécuter une commande en cas d'inégalité. Sans ce dernier le script s'arrête en cas d'inégalité.
sleep qui permet de faire une temporisation afin d'éviter un démarrage de tous les noeuds en même temps et ainsi trop solliciter des ressources limitées.

5) Installer MPICH
(chapitre IV)

6) Répéter toutes les opérations précédentes pour tous les noeuds disponibles

VI) Architecture matérielle

La plupart des composants utilisés sont des PC de bureau récupérés à la déchetterie pour 0 €, il suffit de demander la permission. On peut aussi faire le tour des annonces du boncoin.
Le châssis a été fabriqué à partir d'une étagère en kit.
L'objectif est de fabriquer quelque chose qui ressemble plus ou moins à un serveur lame

1) Fabrication du châssis

Les montants sont des profilés en acier tube carré prépercé

Dimensions: Largeur: 40cm
Profondeur: 45cm
Hauteur: 95cm

2 compartiments verticaux

Le compartiment avant a une profondeur de 27cm et le compartiment arrière de 18cm

Des glissières horizontales sont fixées sur les 2 cotés du compartiment avant espacées de 10cm

Les panneaux sont des plaques de récupération composite alu + polyéthylène

4 ventilateurs disposés en haut du panneau arrière

2) Démontage PC

Avant de démonter le PC il est souhaitable de configurer le bios de la carte mère conformément à son utilisation future, en autre activer le boot réseau et power on sur la perte d'alimentation. Déconnecter tous les périphériques, et les câbles d'alimentation de la carte mère. Ne pas débrancher le câble du refroidisseur de processeur. Démonter soigneusement la carte mère et conserver les vis de fixation

Mettre de coté tous les périphériques qui pourront être utilisés plus tard

Découper le fond de boîtier, sur lequel était fixé la carte mère, aux dimensions des glissières du châssis, de sorte que le panneau arrière de la carte mère puisse se trouver à l'avant du châssis.

3) Premier étage

Installer en bas du châssis dans le compartiment avant une baie de disques qu'on aura récupérée dans un boîtier. On peut ajouter à coté un lecteur CD/DVD, qui peut toujours s'avérer utile

Installer 1 ou 2 disques dans la baie.

Installer une alimentation en bas du compartiment arrière.

Mettre en place le fond de boîtier précédemment découpé sur la première glissière située juste au dessus de la baie de disque et du lecteur CD. Prévoir une fixation de la plaque sur les glissières avant de remettre la carte mère sur son emplacement d'origine.


Connecter les périphériques de stockage ainsi que le câble du panneau frontal. Ce premier noeud servira de serveur dhcp, tftp, nfs, san, sur lequel sera configuré IPXE
Tous les autres noeuds n'auront aucun périphérique de stockage

4) Alimentations

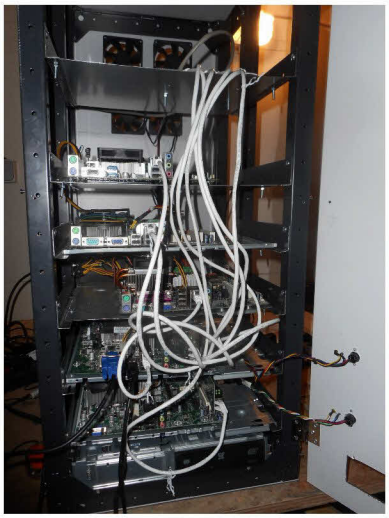
Le premier noeud est alimenté avec une alimentation standard de PC de 280w.
Une deuxième alimentation de 850w est installée au dessus de la première qui alimentera les autres noeuds. Tous les connecteurs ATX 24 broches sont connectés en parallèle ainsi que les connecteurs 4 broches.
Une carte 4 relais (Sain Smart) pilotée par une interface USB avec un petit programme écrit en C permet de démarrer et d'arrêter le cluster en fonction des besoins, afin de réduire la facture EDF et de ne pas contribuer davantage au réchauffement climatique! Il s'agit ici d'une carte de récupération, il existe d'autres dispositifs plus simples d'utilisation sur le marché.

5) Etages supérieurs

Installer les autres cartes mères sur les glissières supérieures avec chacune leur alimentation.
Il peut-être utile de connecter le câble du panneau frontal qui pourra être utilisé pour démarrer une carte mère si le bios n'a pas été correctement configuré au préalable.

6) Réseau

L'interconnexion des systèmes requière au minimum un réseau 1Gb. La configuration des interfaces réseaux devra être optimisée pour prendre en charge des gros débits.
Pour une application nécessitant des gros échanges d'informations comme WRF il faut envisager un réseau type Infiniband, ce qui pourra malheureusement exploser le budget.



7) Exploitation

Avant de lancer un run, le serveur maître, via le programme évoqué au paragraphe 4 commute le premier relais de la carte qui alimente le premier noeud. Puis après une minute d'attente, le temps que le premier noeud soit opérationnel, le programme commute le deuxième relais qui démarre la deuxième alimentation, les autres noeuds vont alors booter via le réseau les uns après les autres.
Après le run le serveur maître envoi via ssh des shutdown séquentiels avec l'option halt pour arrêter proprement les noeuds en respectant un délai d'attente entre chaque commande et dans l'ordre inverse du démarrage, puis commute les relais.