

Building a virtualization cluster based on Xen and iSCSI-SAN

Tutorial Linux Congress 2008 Hamburg

Thomas Groß

teegee

Chemnitz / Germany

thomas.gross@teegee.de

Inhaltsverzeichnis

1 Abstract.....	3
2 The Cluster: Overview.....	4
2.1 Structure of the cluster.....	4
2.2 The cluster servers.....	5
2.3 Virtual machines.....	5
2.4 The iSCSI storage.....	5
2.5 Backup of virtual machines.....	6
2.6 Four levels network architecture.....	7
2.7 High availability.....	8
2.7.1 HA for virtual machines and applications.....	8
2.7.2 HA for the storage.....	8
2.7.3 HA for the cluster.....	8
2.8 Software for virtualization: Xen.....	9
2.9 Management: LAX cluster manager.....	9
3 Preparations.....	11
3.1 The test system.....	11
3.2 Additional software	13
3.3 Openssh autologin.....	13
4 The road to the cluster.....	14
5 Install machines.....	14
5.1 Logical volumes – containers for virtual machines.....	14
5.2 Os templates.....	15
5.2.1 A linux template.....	16
5.2.2 A Windows XP template.....	16
5.3 Install and configure a linux machine from a template.....	17
5.4 Install and configure Windows XP machine from a template.....	17
6 iSCSI-Devices.....	20
6.1 Create iSCSI devices	20
6.2 Import iSCSI devices on a cluster server.....	20
7 XEN.....	22
7.1 Create a Xen control file.....	22
7.2 Manage Xen instances.....	23
7.3 Active and passive virtual machines.....	23
7.4 Manage the Cluster.....	23
7.5 Migrate domains.....	24
8 Conclusion.....	24

1 Abstract

The paper explains in details how to setup and manage a virtualization clusters based on a iSCSI storage and Xen virtualization technology .

A cluster consists of a storage (iSCSI SAN) and 2 or more cluster servers. The SAN is a standard linux server typically having a local raid system and/or external storage connected.

The cluster servers are standard linux servers running a Xen kernel. They need just small local storage for the operating system i.e. they can boot from a flashdisk or a usb memory stick.

The management software laxCluster is a couple of scripts around LVM2, iSCSI, Xen and openssh to handle things like

- manage logical volumes as containers for OS installation
- install a virtual machine from a template into a logical volume
- configure the installation according to your needs
- export / import the logical volumes via iSCSI across a network
- create and deploy Xen control files
- run / stop / migrate / save / restore virtual machines.

Though all this is done by scripts there are a kommander (KDE) based gui tool.

LaxCluster uses the lax infrastructure like the laxDB (openldap) and the openssh autologin channels to remote machines. The lax infrastructure furthermore allows monitoring and notification of cluster servers, virtual machines and services.

LAX is actually based on opensuse 11.0 or SLES10 SP2 and will be available in opensuse build service this autumn.

See also: www.teegee.de -> Downloads -> Vorträge -> Xen-basiertes Cluster mit iSCSI-SAN or xen based cluster with iscsi san.

2 The Cluster: Overview

We do not speak about a computing cluster consisting of hundreds of CPUs for special purposes. This cluster is used for common applications.

Today clusters are typically used because of:

- higher availability of applications
- deliver computing power as necessary
- consolidate physical servers
- better hardware usage
- flexible expansion
- save energy
- easier maintenance

2.1 Structure of the cluster

The cluster consists of a couple of cluster servers, minimum 2, for the computing power and an iSCSI server used as the common storage. Servers and the storage are – at least – simple Linux based machines. The cluster servers and the storage are connected by a separate switch establishing the storage network.

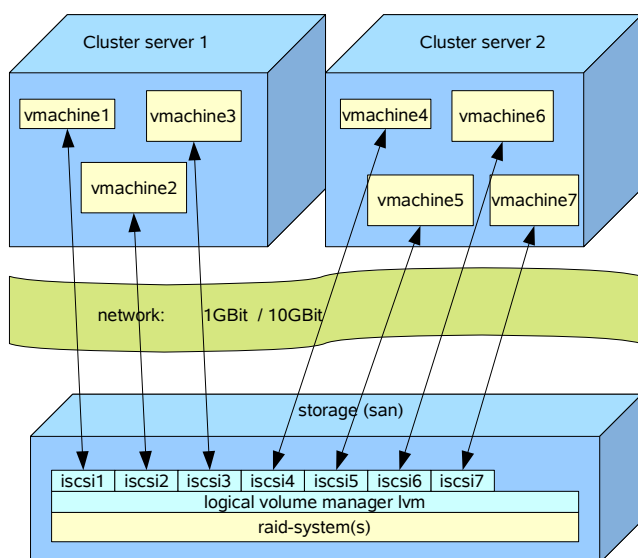


Figure 1: Structure of a cluster.

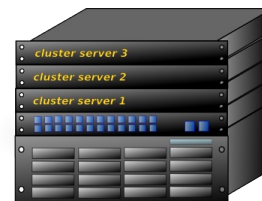


Figure 2: Example of a cluster.

7 units, 12 TB disk, 12 CPUs, 48 Cores, 192 GB memory, 10 Gbit switch

2.2 The cluster servers

The cluster servers deliver the computing power for the virtual machines. They need just a small mass storage for the base operating system only (Xen: dom0) or they can boot via iSCSI.

A cluster server runs the Xen hypervisor kernel based on a standard linux distribution, e.g. opensuse 11.0 or SLES10 SP2. Furthermore it needs a special software package (opensuse: open-iscsi) to connect to the iSCSI storage server. It imports (it's called "login") the iSCSI resources from the storage server and starts the operating systems, installed in these resources, as virtual machines.

2.3 Virtual machines

The virtual machines run on the cluster servers using their memory and CPU. Because they are booted via the Xen dom0 from iSCSI resources they "see" only SATA /IDE disks – typically as /dev/sda, /dev/hda or C:. Additional disk resources e.g. for application data (/data, D:) are possible and must be named in the Xen control file.

Because all cluster servers are based on one iSCSI storage migration, even life migration, of a virtual machine to another cluster server is possible. In case of life migration the cluster servers hardware architecture must be identical (AMD-AMD, Intel-Intel).

2.4 The iSCSI storage

The storage server runs a standard Linux OS too. It especially needs a iSCSI target software package (opensuse: iscsitarget). The mass storage is typically built from a internal RAID system but can be extended e.g. by external fibre channel storage systems. This way it realizes the architecture

disk → raid → lvm → iscsi

The logical volume manager (lvm) integrates one or more RAID systems to one volume group. Then logical volumes are created to install the operating systems into or to hold additional user/application data. These logical volumes (i.e. their block devices) are published to the storage network as iSCSI devices.

Because of usage of LVM it's easy to increase or shrink the virtual machines.

The storage server can run a Xen hypervisor too e.g. to run virtual machines for maintenance.

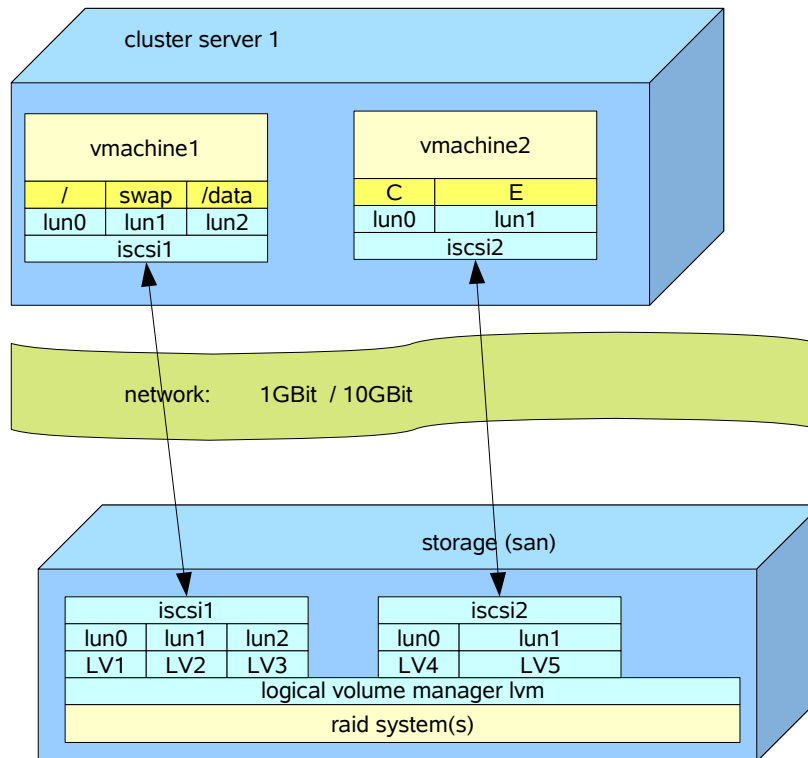


Figure 3: Cluster structure more detailed

2.5 Backup of virtual machines

The virtual machines can be saved online by the LVM snapshot technology and by storing in a simple compressed tar archive.

The restore process has to create logical volumes, unpack the tar archive into that volumes and publish them via iSCSI. For easier restore we store the Xen control file too. By the restore process limitations such as file system size and memory usage can be changed.

2.6 Four levels network architecture

The strict usage of the technology leads to the 4 levels network architecture which are

- the clients, connect by the intranet to the virtual machines (and their applications)
- the cluster servers which run the virtual machines
- the iSCSI storage where data are located
- the backup systems to save data from the iSCSI storage

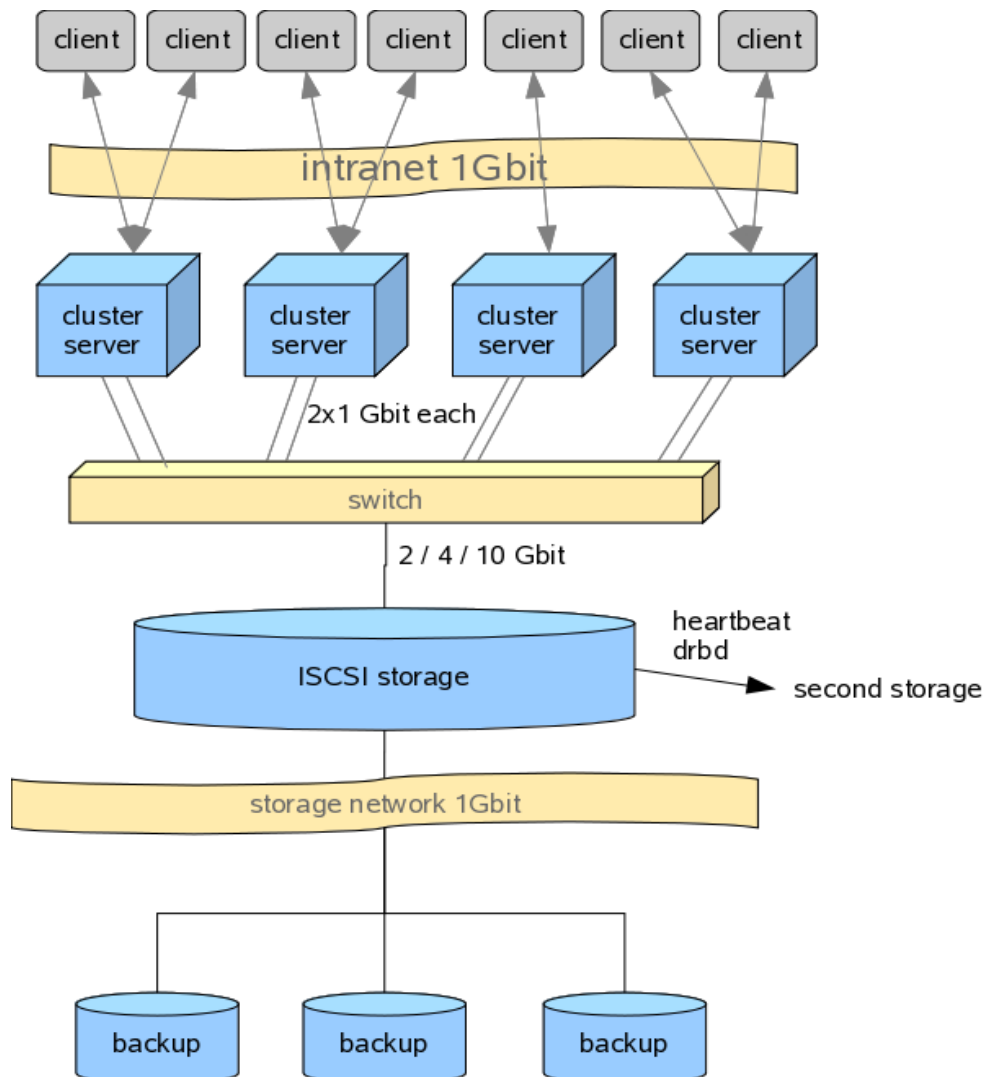


Figure 4: 4 levels architecture

These 4 levels are connected by 3 separate networks which optimize network throughput by separating user interactions from storage tasks.

This architecture offers a long term upscale strategy by adding / replacing cluster servers and furthermore for the development of the network, the storage and the backup system.

The applications benefit from newer, faster hardware, because their virtual machines can be copied or migrated from old cluster servers to new ones.

2.7 High availability

2.7.1 HA for virtual machines and applications

HA for applications can be realized by restarting the associated virtual machine on another cluster server. So it's necessary to watch the state of the cluster servers, the virtual machines and the applications.

If a cluster server or a virtual machine fails the machines and their data are still available on the iSCSI storage so they can be restarted on another cluster server.

The automated restart of virtual machines on other cluster servers after a crash is a complicated task. Probably the available cluster servers must be reorganized by shrinking their virtual machines because of lack of memory. This can be done by heartbeat2. teegee is working on a solution based on laxMonitor.

2.7.2 HA for the storage

For real HA you need of course a second iSCSI storage. The storages have to be connected by a high speed link for real time data exchange. The data exchange can be realized by distributed block device (DRBD). DRBD organizes a real time data duplication across a network link. This way all data of san1 are duplicated to san2 and vice versa. The monitoring of the storage servers handles heartbeat1.

2.7.3 HA for the cluster

But high availability is more than setting up the technologies to keep the machines online. You should also think of location security. Place the clusters / cluster servers / storage servers in separate locations in case of destruction of one location.

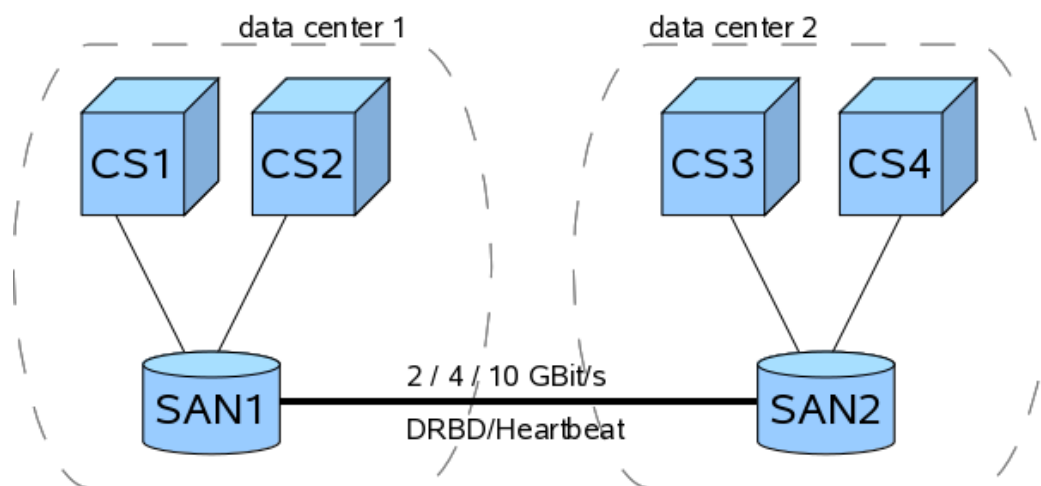


Figure 5: High available cluster

2.8 Software for virtualization: Xen

We use the Xen hypervisor as part of the opensuse11.0 / SLES10 SP2 distributions (Xen 3.2).

Xen runs on each linux compatible hardware, is free (GPL) software and is well supported by major industry vendors. It supports both the virtualization hardware extension of AMD and Intel CPUs, i.e. it runs unmodified operating systems (Windows).

Xen supports the exclusive usage of hardware components by virtual machines. This is e.g. necessary to realize firewall machines.

Furthermore Xen supports multiple internal bridges and internal subnets by NAT / routing so it offers the possibility to realize whole network structures in one machine.

Xen is managed in Unix style. It uses some commands (xm ...) and text based configuration files. Detailed informations of the state of the virtual machines are available.

2.9 Management: LAX cluster manager

Though all task to setup and manage virtual machines can be done manually (shell commands) this could be at least a hard and tiresome job. That's why we developed scripts which bundle the technologies LVM, iSCSI and Xen to simple commands. These commands

- install virtual machines from templates and customize them
- manage active and inactive virtual machines
- save / restore virtual machines

Here are some examples:

```
vman-install [-i ip-address] [-m netmask] [-g gateway] [-a mac-address] [-n nameserver] [-r memory] [-f disksize] [-s swappsize] [-x filesystem] [-c vcpus] [-d description] [-t shutdown_time] cluster template newdomu
```

```
vman-start cluster domu [clusterserver]
```

```
vman-restore cluster domu [backup]
```

The LAX cluster scripts can manage more than one cluster.

The LAX software e.g. resides on the storage server. The communication between the storage server and the cluster servers needs openssh autologin channels. This way remote activities like starting a virtual machine can be activated from the storage server.

New virtual machines are automatically integrated to the LAX network database (openldap) so they are available for additional task like inventory or monitoring.

There is a simple kommander based graphical interface available too – but today in german language only (sorry).

The screenshot displays the LAX Cluster Manager interface. At the top left, a 'Cluster-Baum' (Cluster Tree) shows a hierarchy with 'cl1' containing 'cs1' and 'cs2'. The main area is divided into two panes. The right pane, titled 'Eigenschaften Clusterserver', shows server details: 'freier Speicher' (free memory) at 1361 MByte, 'CPU' with 2 cores at 2493 MHz, and 'Plattenplatz (SAN)' at 165.34G. Below this, a row of controls for 'virtuelle Maschinen' (virtual machines) includes buttons for Stop, Kill, Move, Defs, New, and Con. The bottom section, 'aktive virtuelle Maschinen', contains a table with columns for Maschine, Cserver, ID, Speicher, vCPU, Status, Zeit, Disk, MAC, IP-Adresse, Template, and Info. The table lists several VMs with their respective configurations and statuses.

Maschine	Cserver	ID	Speicher	vCPU	Status	Zeit	Disk	MAC	IP-Adresse	Template	Info
etch1	cs2	1	512	1	-b----	5.1	6G	00:16:3E:5E:7C:B1	192.168.30.204	debian-etch	Beschreibung_erererer
p3	cs1	1	128	1	-b----	4.5	4G	00:16:3E:A1:2F:9A	192.168.30.240	debian-etch	Beschreibung_p3
probe1	cs2	8	128	1	-b----	0.0	4G	00:16:3E:EC:70:3B	192.168.30.241	suse103	probe1
probe3	cs1	3	1024	1	-b----	4.3	8G	00:16:3E:32:69:07	192.168.30.197	debian-etch	probe3
test1	cs2	2	256	1	-b----	8.3	4G	00:16:3E:31:DB:5D	192.168.30.201	suse103-update	test1
test2	cs1	4	128	1	-b----	7.5	4G	00:16:3E:75:75:39	192.168.30.202	suse103-update	test2
test3	cs1	5	128	1	-b----	7.3	4G	00:16:3E:B7:5F:62	192.168.30.203	suse103-update	testmaschine_3
ttt	cs2	3	256	1	-b----	4.8	4G	00:16:3E:B6:0F:CE	192.168.30.205	debian-etch	Beschreibung_ttt
winxp1	cs2	7	256	1	-b----	7101.5	6G	00:16:3E:10:D1:B7	192.168.30.211	wxp	Beschreibung_winxp1
winxp2	cs1	6	512	1	r-----	6356.3	4G	00:16:3E:2D:94:55	192.168.30.212	wxp	Windows Testmaschine2

Figure 6: LAX: cluster manager

XEN: virtuelle Maschine anlegen

Cluster: **cl1**

VM-Name: Buchstaben, Ziffern, _ , keine Leerzeichen wird Hostname

Vorlage: Betriebssystem

OS: Linux

Beschreibung:

IP-Adresse:

Netzmaske:

Gateway:

Nameserver:

MAC-Adresse: MAC-Adresse: leer: automatisch vergeben

Plattengröße: z.B. 30G

Swap-Größe: i.A. doppelter Hauptspeicher; z.B. 1G, 512M

Hauptspeicher: in MByte, z.B. 512 oder 2048

Figure 7: LAX: install a new virtual machine

3 Preparations

3.1 The test system

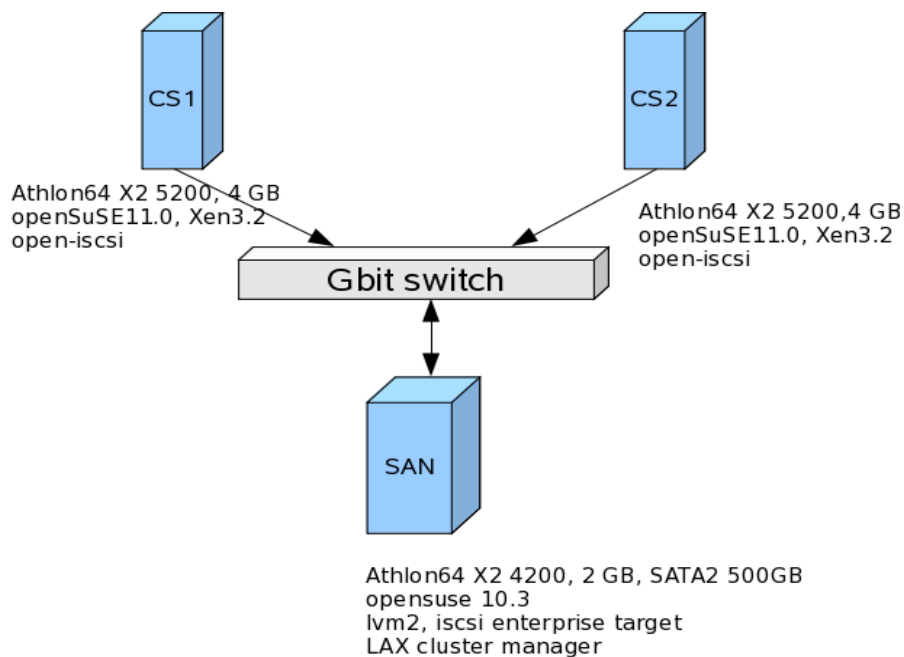


Figure 8: The test system consist just of simple pc-like computers.

We use the opensuse 10.3 and 11.0 distribution. Because we mostly work on the storage server we install the standard kde3 desktop and optional a NX server if you plan to work remote there.

The cluster servers need no gui.

The switch is a simple Gbit switch here.

Put all the system names in /etc/hosts for name service independent name resolution, e.g.

```
192.168.30.21    san1.klasse.kurs    san1    # iscsi storage
192.168.30.22    cs1.klasse.kurs     cs1     # cluster server 1
192.168.30.23    cs2.klasse.kurs     cs2     # cluster server 2
```

3.2 Additional software

on cluster servers install

- The Xen kernel and related software for managing the hypervisor and virtual machines
- open-iscsi, the package for importing (“login to”) iSCSI-devices

on the storage server install

- iscsitarget, FreeNX and NX
- kpartx to find the partition inside a Windows XP image
- lvm2 ist already installed

3.3 Openssh autologin

Login as root and create a personal openssh key:

```
# ssh-keygen          # press <enter> to all questions; no passphrase
```

Now you have 2 files in /root/.ssh: *id_rsa.pub* and *id_rsa*.

Distribute the public key to the cluster servers:

```
# ssh-copy-id -i .ssh/id_rsa.pub root@c11
# ssh-copy-id -i .ssh/id_rsa.pub root@c12
```

You can login now to a cluster server without typing a password; e.g:

```
# ssh root@c11
```

To have a shortcut for running a program on a cluster server e.g. put these lines to /root/.profile:

```
runcl1() { ssh root@c11 $@; }
runcl2() { ssh root@c12 $@; }
```

Later we also need openssh autologin channels (root-root) between the cluster servers.

4 The road to the cluster

To have a cluster working we have to

- install virtual machines on the storage and prepare them as iscsi devices
- run the machines as Xen virtual machines on the cluster servers

These are the broad step to setup a (linux) machine

1. create empty logical volumes for the root filesystem and for the swap
2. copy a linux template to the root filesystem's lv
3. configure the new installed system (hostname, network)
4. create an iSCSI device for the machine and “publish” it
5. create a Xen control file for the machine and copy it to both machines
 - on one cluster server -
6. import (“login to”) the associated iSCSI device
7. start the virtual machine
8. pray xensource.org ;-)

5 Install machines

5.1 Logical volumes – containers for virtual machines

We decided to use logical volumes – not images – to hold our os installations. The advantages of lvs vs. images are

- easy to handle, increase and shrink
- “natural” software level on top of the disks (raids)
- can be copied (saved) online via snapshot-technology

The available disk storage is hold in one volume group (vg). Then the logical volumes (lv) can be created from the volume group. If necessary you can enlarge the vg later by adding new physical devices or storage subsystems.

Example:

Say we have a RAID5 system of 2 TByte capacity represented in the operating system as **/dev/raid**. Now let's create a volume group and add the raid to the vg.

```
vgcreate system /dev/raid
```

creates the vg "system" and puts the physical volume "/dev/raid" into. If you add later a second storage available as /dev/extraid you can extend the vg "system" as

```
vgextend system /dev/extraid
```

The command

```
vgdisplay [vg_name]
```

shows details of the vg, especially the size of allocated and free storage.

Now let's create logical volumes. Logical volumes are represented in the system as block devices (similar to partitions). But their size can be easily changed.

```
lvcreate -L 10G -n pc46 system
```

Logical volume "pc46" created

creates the lv "pc46" from the vg "system". The associated size is 10 GBytes. The corresponding block device is "/dev/system/pc46". But use the mapped name **/dev/mapper/system-pc46** for further operations .

The following command removes it. Because of '-f' (force) a question is suppressed.

```
lvremove -f /dev/system/pc46
```

Be sure to name your lvs equivalent to the machine names you want to install there later. This helps you not to confuse by a growing number of lv's:

machine	lv_name	block device	mount-point
pc46	pc46-root	/dev/mapper-pc46-root	/
	pc46-swap	/dev/mapper-pc46-swap	swap
	pc46-data	/dev/mapper-pc46-data	/data
xp1	xp1-root	/dev/mapper-xp1-root	C:
	xp1-d	/dev/mapper-xp1-d	D:

This concept isn't a must. But if you don't organize such a naming scheme you have to register your machines, lvs and filesystems in a separate table.

5.2 Os templates

We place os templates in a one directory structure, e.g. in /vmtemplates:

```
/vmtemplates      # maybe a mount point for a logical volume
opensuse11.0-base # directory for a special os
  installation    # the a copy of the filesystem
  bin            # config-scripts special for this os
  info          # an information about the os
  vm            # a template for the Xen control file
```

5.2.1 A linux template

At least a linux template is a copy of an existing installation except the directories /sys, /proc and /tmp, which are needed as empty directories. Say we have updated our machine pc46 (not running) and want to create a new template “opensuse11.0-update”:

```
mkdir /tmp/xenmount                # mount point
mount /dev/mapper/system-pc46 /tmp/xenmount # mount the lv of the machine
mkdir /vmtemplates/opensuse11.0-update  # create the template structure
mkdir /vmtemplates/opensuse11.0-update/installation
cp -ax /tmp/xenmount/* /vmtemplates/opensuse11.0-update/installation # copy
cd /vmtemplates/opensuse11.0-update/installation/tmp; rm -r * # empty /tmp
```

Because the machine isn't running the directories /proc and /sys are empty.

A very new os is typically installed by yast2. It supports the installation from a CD/DVD very well.

Here is a template for the xen control file:

```
disk = ['phy:/dev/system/VM-NAME-root,sda,w', 'phy:/dev/system/VM-NAME-swap,sdb,w' ]
memory = VM-RAM
vcpus = VM-VCPU
builder = 'linux'
name = 'VM-NAME'
vif = [ 'mac=VM-MAC' ]
localtime = 0
on_poweroff = 'destroy'
on_reboot = 'restart'
on_crash = 'restart'
extra = ' TERM=xterm'
# This is a bootloader used to boot paravirtualized domains (kernel from the domU)
bootloader = '/usr/lib/xen/boot/domUloader.py'
bootargs = '--entry sda:/boot/vmlinuz-xen,/boot/initrd-xen'
# Verwendung des Kernels aus /boot-Verzeichnis der dom0
# die dazu passenden Module muessen im Gastsystem bzw. gleich im Template in /lib/modules/ existieren
#kernel = '/boot/vmlinuz-xen'
#ramdisk = '/boot/initrd-xen'
#root = '/dev/hda ro'
#os=Linux
```

Variables like “VM-NAME” have to be replaced by the real values.

We use the domU-loader feature, i.e. the virtual machine uses it's own xen kernel.

5.2.2 A Windows XP template

The Windows XP isn't a simple copy of the file system.

```
# ls -lh
-rw-rw---- 1 root root 4,0G 29. Jan 2008 wxp.img
# file wxp.img
wxp.img: x86 boot sector, Microsoft Windows XP MBR (german), Serial 0xb518b518
```

This image is a “dd copy” of a Windows XP installation which contains a **sysprep** installation.

5.3 Install and configure a linux machine from a template

We create the necessary logical volumes for a virtual machine, e.g. for “mytest”:

```
lvcreate -L 10G -n mytest-root system
lvcreate -L 1G -n mytest-swap system
mkfs -t ext3 /dev/system/mytest-root
```

Now let's copy the files from the template to the lv:

```
mkdir /tmp/xenmount 2>/dev/null
mount /dev/system/mytest-root /tmp/xenmount
cp -ax /vmtemplates/opensuse11.0-base/* /tmp/xenmount
```

The next step configures the new installed machine, i.e. sets hostname and network-data. This depends of the template system. The configuration of a suse system is slightly different from a debian.

At a suse system configure

```
/etc/hosts
/etc/HOSTNAME
/etc/resolv.conf # nameserver
/etc/sysconfig/network/ifcfg-eth0
/etc/sysconfig/routes
```

At a debian system configure

```
/etc/hosts
/etc/hostname
/etc/mailname
/etc/resolv.conf
/etc/network/interfaces
```

Then unmount the configured system.

```
umount /tmp/xenmount
```

If you run a xen kernel on the storage server you can now create a xen control file, place it in /etc/xen/vm and start the virtual machine from the lvm.

5.4 Install and configure Windows XP machine from a template

Windows can not be installed and run from a logical volume directly. It needs a partition. That's why we need to find the partition in the template.

Example: the new machine is myxp.

```
# lvcreate -L 10G -n myxp-root system
Logical volume "myxp-root" created
```

At first copy the image to the prepared (empty) lv.

```
# dd if=/vmtemplates/wxp/installation/wxp.img of=/dev/system/myxp-root
8388608+0 records in
8388608+0 records out
4294967296 Bytes (4,3 GB) copied, 497,581 s, 8,6 MB/s
```

Now we have to change (extend) the partition borders. The dd copy sets the end of the partition to about 4GB but we have 10GB. We have just one Partition.

```
fdisk /dev/system/myxp-root <<EOF &> /dev/null
d                # delete the partition
n                # create a new partition
p                # a primary partiton
1                # number one
                # first cylinder
                # last cylinder
t                # toggle type of the partition
7                # ntfs type
a                # set active (bootable) partition
1                # partition one
w                # write to disk
EOF
```

Next we want get a handle for the partition inside. We need kpartx.

```
# free_loop=$(losetup -f) &> /dev/null      # the next free loop device
# echo $free_loop
/dev/loop0
# losetup /dev/loop0 /dev/system/myxp-root # connects /dev/loop0
```

Now kpartx gives us a block device to the partition(s) inside

```
# kpartx -l /dev/loop0                    # just see what we get
loop0p1 : 0 20964762 /dev/loop0 63
# kpartx -a /dev/loop0                    # creates the block device /dev/mapper/loop0p1
```

Resize s the ntfs file system to the size of the lv

```
# ntfsresize -f /dev/mapper/dev/loop0p1
ntfsresize -f /dev/mapper/loop0p1
ntfsresize v1.13.1 (libntfs 9:0:0)
Device name      : /dev/mapper/loop0p1
NTFS volume version: 3.1
Cluster size    : 4096 bytes
Current volume size: 4285338112 bytes (4286 MB)
Current device size: 10733958144 bytes (10734 MB)
New volume size  : 10733953536 bytes (10734 MB)
Checking filesystem consistency ...
100.00 percent completed
Accounting clusters ...
Space in use     : 1925 MB (44,9%)
Collecting resizing constraints ...
Schedule chkdsk for NTFS consistency check at Windows boot time ...
Updating $BadClust file ...
Updating $Bitmap file ...
Updating Boot record ...
Syncing device ...
Successfully resized NTFS on device '/dev/mapper/loop0p1'.
```

The next step prepares the Windows XP sysprep mechanism

```
# mkdir /tmp/xenmount/myxp-root          # empty dir for mounting the partiton
# mount -t ntfs-3g /dev/mapper/loop0p1 / tmp/xenmount/myxp-root
```

Here is a part of a sysprep.inf, the control file for setting up Windows XP at the first boot. Especially edit the hostname, the network parameters and the username.

```
# cat /tmp/xenmount/myxp-root/Sysprep/sysprep.inf
;SetupMgrTag
[Unattended]
    OEMSkipEula=Yes

[GuiUnattended]
    AdminPassword="password"
    EncryptedAdminPassword=NO
    OEMSkipRegional=1
    TimeZone=110
    OEMSkipWelcome=1

[UserData]
    ProductID=XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
    FullName="teegee"
    OrgName="teegee"
    ComputerName=pc1

[params.MS_TCPIP]
    DNS=No
    UseDomainNameDevolution=No
    EnableLMHosts=Yes
    AdapterSections=params.MS_TCPIP.Adapter1

[params.MS_TCPIP.Adapter1]
    SpecificTo=Adapter1
    DHCP=No
    IPAddress=192.168.100.250
    SubnetMask=255.255.255.0
    DefaultGateway=192.168.100.1
    DNSServerSearchOrder=192.168.100.100
    WINS=No
    NetBIOSOptions=0
[sysprepcleanup]
```

Finally umount the partition, delete the kpartx block devices and detach the loop-device.

```
# umount /tmp/xenmount/myxp-root
# rmdir /tmp/xenmount/myxp-root
# kpartx -d /dev/loop0
# losetup -d /dev/loop0
```

The first start of our myxp virtual machine takes about 4 minutes because of preparing the system.

6 iSCSI-Devices

iSCSI-devices are handled by the *ietd*-daemon, which is part of the *iscsitarget* package. Check if it runs:

```
rciscsitarget status
Checking for iSCSI target service          running
```

`rciscsitarget` is a link to the init-script `/etc/init.d/iscsitarget` (SuSE-style).

6.1 Create iSCSI devices

The `iscsi`-devices are configured in `/etc/ietd.conf`. The file starts with a global section defining the behavior of the *ietd* daemon. Then a list of target definition follows, e.g.:

```
Target iqn.san1.my.domain:mytest
Lun 0 Path=/dev/system/mytest-root,Type=fileio
Lun 1 Path=/dev/system/mytest-swap,Type=fileio
```

A target line starts with the keyword “Target”. Then follows the name of the target, which has to be built by “`iqn.<fqhn>:<free_name>`”.

The next lines describe the logical units included in the target. Place here all the logical volumes you want to use in the virtual machine. In case of our “mytest” we use the `lvs` for the root filesystem and the swap filesystem.

“Path” specifies the path to the block device. The “Type=fileio” defines the default io scheme. Use *Type=blockio* if you need highest performance and have fast hardware.

To make the new `iscsi` target available reload/restart the *ietd* daemon:

```
rciscsitarget restart
```

The *ietadm* command allows to add targets and luns too, but they are not written to the *ietd.conf* for permanent usage..

You can see what devices are defined

```
cat /proc/net/iet/volume
```

and which are active

```
cat /proc/net/iet/session
```

6.2 Import iSCSI devices on a cluster server

The client side is called “initiator”. No special daemon is needed but install the package *open-iscsi*. “Import” the `iscsi` targets in 2 steps:

First discover what targets are available:

```
iscsiadm -m discovery -I default -t st -p 192.168.30.21
-m discovery          mode
-I default            default interface
-t st                 type: send targets
-p IP[:port]         portal (iscsi server)
```

The found nodes are available in `/etc//iscsi/nodes`.

Then connect the target you want:

```
iscsiadm -m node -I default -T iqn.san1.klasse.kurs:mytest -p 192.168.30.21 -l
    -m node                mode
    -I default             default Interface
    -T target
    -p IP[:port]          portal (iscsi server), default port is 3260
    -l                    login
```

Now the corresponding iscsi luns are available as

```
/dev/disk/by-path/ip-192.168.30.21:3260-iscsi-iqn.san1.klasse.kurs:mytest-lun-0
/dev/disk/by-path/ip-192.168.30.21:3260-iscsi-iqn.san1.klasse.kurs:mytest-lun-0
```

These are the lock devices we include in the Xen control file to give the virtual machines their disk resources.

The luns are also available by id (*/dev/disk/by-id/*) → scsi-id's, but this is only useful if we define the scsi-id in the ietd.conf. We didn't, so it's easier for us to use */dev/disk/by-path/* were the names correspond with our machine names.

For better security the login to the iscsi device can be checked by a username/password pair. Put such an additional line to your target definition:

```
IncomingUser peter peters_password
```

This can be useful if the iscsi server is available from the users side. In our network scheme the iscsi targets are available by the cluster servers only (across the storage network) .

7 XEN

The Xen hypervisor (the Xen kernel) is already running after the installation of the appropriate packets and a reboot. Enable the xen daemon to start at boot time:

```
# rcrcxend start           # starts xend now
# chkconfig -a xend       # integrates xend into system boot
```

The system where you login is called domain0 or dom0. From here the virtual machines are managed.

Maybe you want to limit the memory the dom0 uses. That helps to count the memory available. Edit the corresponding line in `/boot/grub/menu.lst`.

```
kernel /xen.gz dom0_mem=512M
```

If xend is running the standard bridge `xenbr0` was set up too.

```
# brctl show
bridge name      bridge id          STP enabled      interfaces
xenbr0           8000.feffffffffff no                vif0.0
                                                         peth0
```

Thereby the hardware interface was renamed to `peth0` and the dom0's interface `eth0` has now its data, but `eth0` is connected via `vif0.0` to the bridge.

7.1 Create a Xen control file

We already have an installation in a lv and the iscsi device on a cluster server available. Now we have to create a xen control file for our machine. We use the file `vm` as template from the `template` directory. We just have to fill up the correct values and copy the file to both cluster servers into `/etc/xen/vm`.

```
disk = [ 'phy:/dev/disk/by-path/ip-192.168.30.21:3260-iscsi-iqn.san1.klasse.kurs:os11-base-lun-0,sda,w',
'phy:/dev/disk/by-path/ip-192.168.30.21:3260-iscsi-iqn.san1.klasse.kurs:os11-base-lun-1,sdb,w' ]
memory = 512
vcpus = 1
builder = 'linux'
name = 'os11-base'
vif = [ 'mac=00:16:3E:83:5A:EA' ]
localtime = 0
on_poweroff = 'destroy'
on_reboot = 'restart'
on_crash = 'restart'
extra = ' TERM=xterm'
bootloader = '/usr/lib/xen/boot/domUloader.py'
bootargs = '--entry sda:/boot/vmlinuz-xen,/boot/initrd-xen'

# die dazu passenden Module muessen im Gastsystem
# beziehungsweise gleich im Template in /lib/modules/ existieren
#kernel = '/boot/vmlinuz-xen'
#ramdisk = '/boot/initrd-xen'
#root = '/dev/hda ro'
#os=Linux
```

7.2 Manage Xen instances

Having the xen daemon and the network bridge running and the control file `/etc/xen/vm/mytest` start the machine (as root) by

```
xm create mytest
```

Maybe watch for possible problems:

```
tail -f /var/log/xen/xend.log
```

Look as your machine runs:

```
xm list
```

Stop it again:

```
xm shutdown mytest
```

Show details about the hypervisor, e.g. the available memory

```
xm info | grep max_hvm_memory
max_hvm_memory      : 1593
```

7.3 Active and passive virtual machines

You can install many machines but run only a few of them. This is useful for testing purposes but remember which machines you have installed and what are their parameters. Xen doesn't help you here, it just cares of the running machines.

Furthermore Xen just cares of the machine itself, not what they do or how they are configured. But a administrator should know that mytest is an opensuse11.0, what IP it uses or what's its purpose in your network.

7.4 Manage the Cluster

Managing the cluster means that you place the machines to different cluster servers. Be sure to have always more memory available than noticed in the xen control file of your virtual machine:

```
memory = 512
```

otherwise the machine doesn't start.

You can have a look at

```
xm top
```

to what domains need more computing power.

If you have to shrink the memory a machine uses try

```
xm mem-set mytest 400 # megabyte
```

7.5 Migrate domains

It is possible to move a machine from one cluster server to another by *xm migrate*, if the cluster servers use one common storage – as the iscsi server.

Migrate *mytest* to cluster server *cs1* (if it runs on *cs2*).

```
xm migrate mytest cs1
    -l          live migration
    -r Mbit/sec maximum transfer rate
```

The migration process cuts the connection to the domain a short time. Live migration wants to minimize that cut to milliseconds only. At least, migrate freezes the domain, copies the memory to the other cluster server and releases it there.

8 Conclusion

Setup and management of a Xen based virtualization cluster with a iSCSI storage is possible with standard Linux means and with standard pc hardware components. Hereby Linux runs both on the cluster servers and the storage server. Using a distribution with Xen hypervisor included is recommend.

For better performance on Windows systems use SLES10 SP2 and the additional driver pack from Novell.

The LAX software helps you to manage the complicated tasks by bundling the different technologies to single commands. Furthermore it integrates the virtual machines into the LAX database for other purposes.

References

[1] Henning Sprang, Timo Benk, Jaroslaw Zdrzalek, Ralph Dehner: Xen Virtualisierung unter Linux, Open Source PRESS, 2007

[2] <http://www.xen.org/>

The LAX software and scripts were developed from the teegee people namely Thomas Groß, Rene Kunze, Elke Humml, Thorsten Ulrich, Sirko Kemter and Bernd Stolle

Windows and Windows XP is a registered trademark of Microsoft Corporation in the United States and other countries.