Heart attack Virtualization

An HA virtualization cluster based on Pacemaker, Corosync, Xen and DRBD

Davide Vaghetti davide.vaghetti@ing.unipi.it
Centro Servizi Informatici Facoltà di Ingegneria
Università di Pisa

The Challenge

Building a new server farm for the Computing Center of the Faculty of Engineer

Targets

hardware usage consolidation
Physical servers should host multiple virtual instances. Existing servers will be converted to virtual instances.

high available servers
In the event of an hardware failures, virtual instances will be relocated to a different physical node.

decoupling servers deployment
Servers deployment should be decoupled from hardware set up and operative system installation.

simplify servers management
It should be possibile to manage startup/shutdown/restart operations of each server through a single choerent console. The same apply to filesystem management and backup operations.

Virtualization platform requirements

hardware access
virtual instances must have directly access hardware (PCIs, NICs, HDs, etc.)

flexibility
it must be possible to assign different resources to each virtual instance

performances
virtualized servers should reach near-metal performances

reliability
the virtualization platform should implement high availability technologies

management
the less resources to dedicate to management, the better

The Computing Center of the Faculty of Engineer

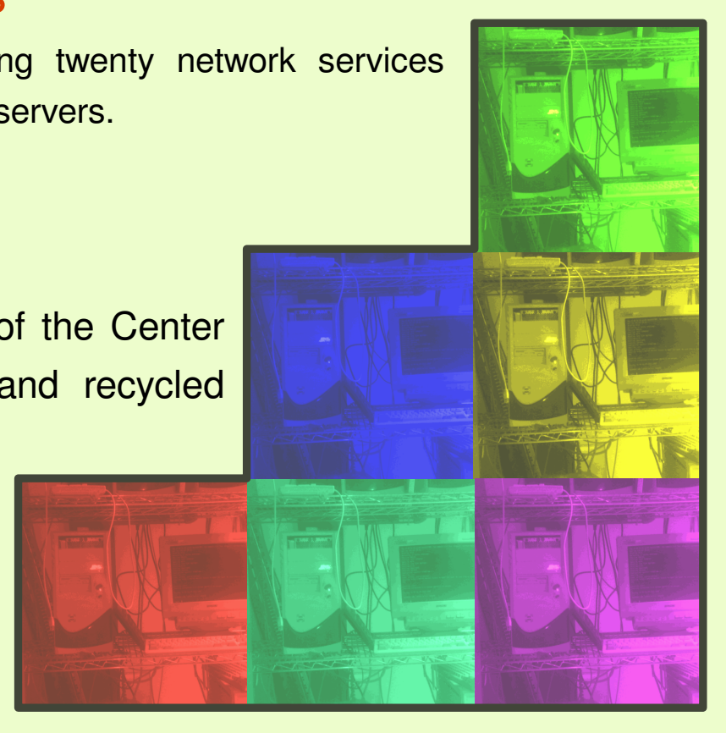
7000 users
The Computing Center of the Faculty of Engineer (CSIFI) serves a large community composed by students, professors, researchears, technicians and administrative workers.

manages the network
The area of the Faculty of Engineer comphrehends six buildings, ten departments, nine students room, one library. Each structure has its own subnet. Local networks are managed by a local admin; network authentication, routing and the physical network backbone are managed by CSIFI.

provides network services
The center provides all the main network services for the engineers community: authentication, mail, dns, web, vpn, e-learning, video conferencing, etc.

The 2009 server farm

tens of services = tens of servers
Each service had its own server, and running twenty network services meaned having to manage the same amount of servers.



obsolete hardware
During the years, the computing facilities of the Center --- mainly based on desktop platforms and recycled hardware running FreeBSD and Debian GNU/Linux --- grew old. Moreover, the server hardware was all but uniform, ranging from ancient Pentium III to Intel Xeon.

The Solution

Materials

Hardware

vnode1	Dual Quadcore Xeon - 16 Gbyte RAM 4 x 146 Gbyte SAS 15K rpm HDs
vnode2	Dual Quadcore Xeon - 16 Gbyte RAM 4 x 146 Gbyte SAS 15K rpm HDs
vnode3	1 x Quadcore Xeon - 8 Gbyte RAM 2 x 320 Gbyte SATA 7.2K rpm HDs
vstor1	1 x Dual Dualcore Opteron - 4 Gbyte RAM 2 x 2 Tbyte SATA 7.2K rpm HDs 4 x 500 Gbyte SATA 7.2K rpm HDs

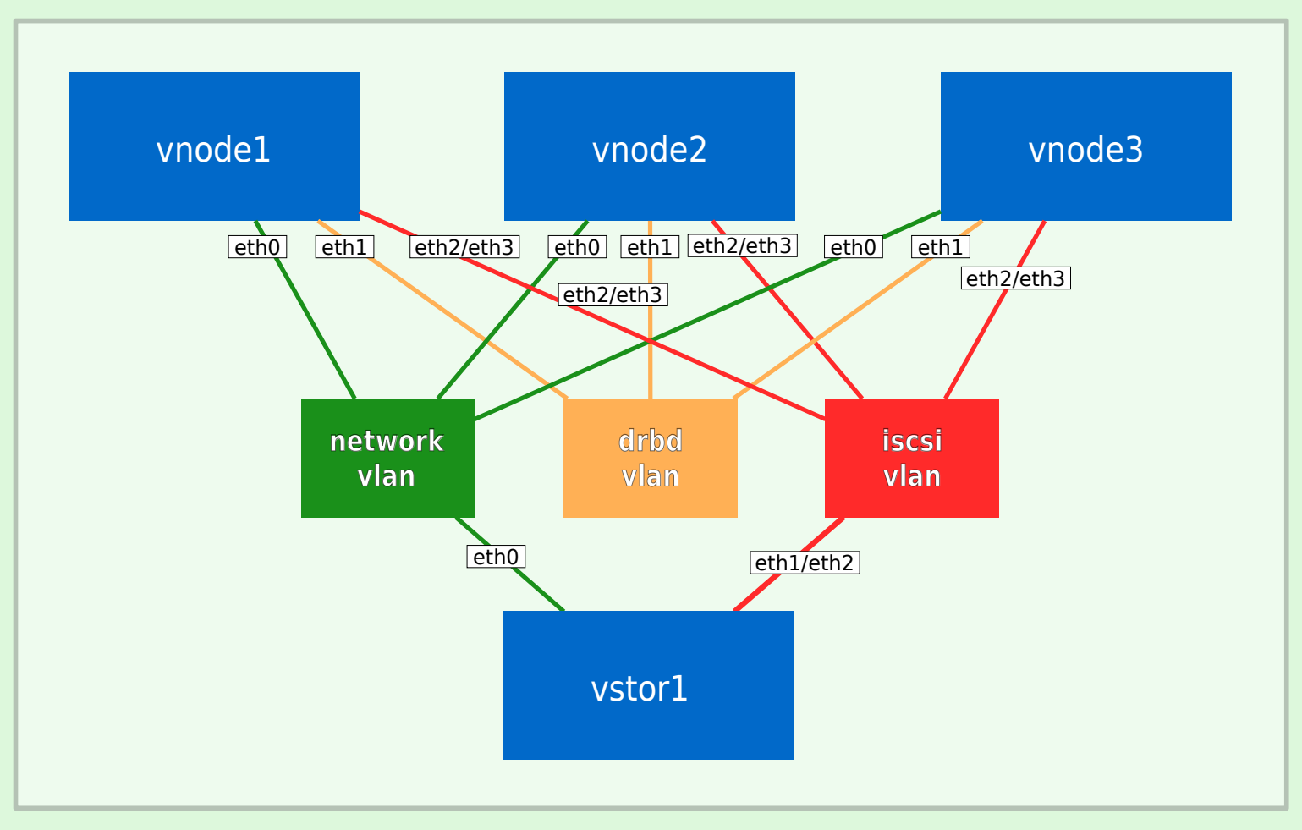
Software

- Debian GNU/Linux Lenny
- Linux Kernel 2.6.26-2-xen-amd64
- Xen 3.2.1
- Corosync 1.2.1 (cluster engine)
- Pacemaker 1.0.9 (cluster resource manager)
- DRBD 8.3.7
- LVM 2

Cluster Network Layout

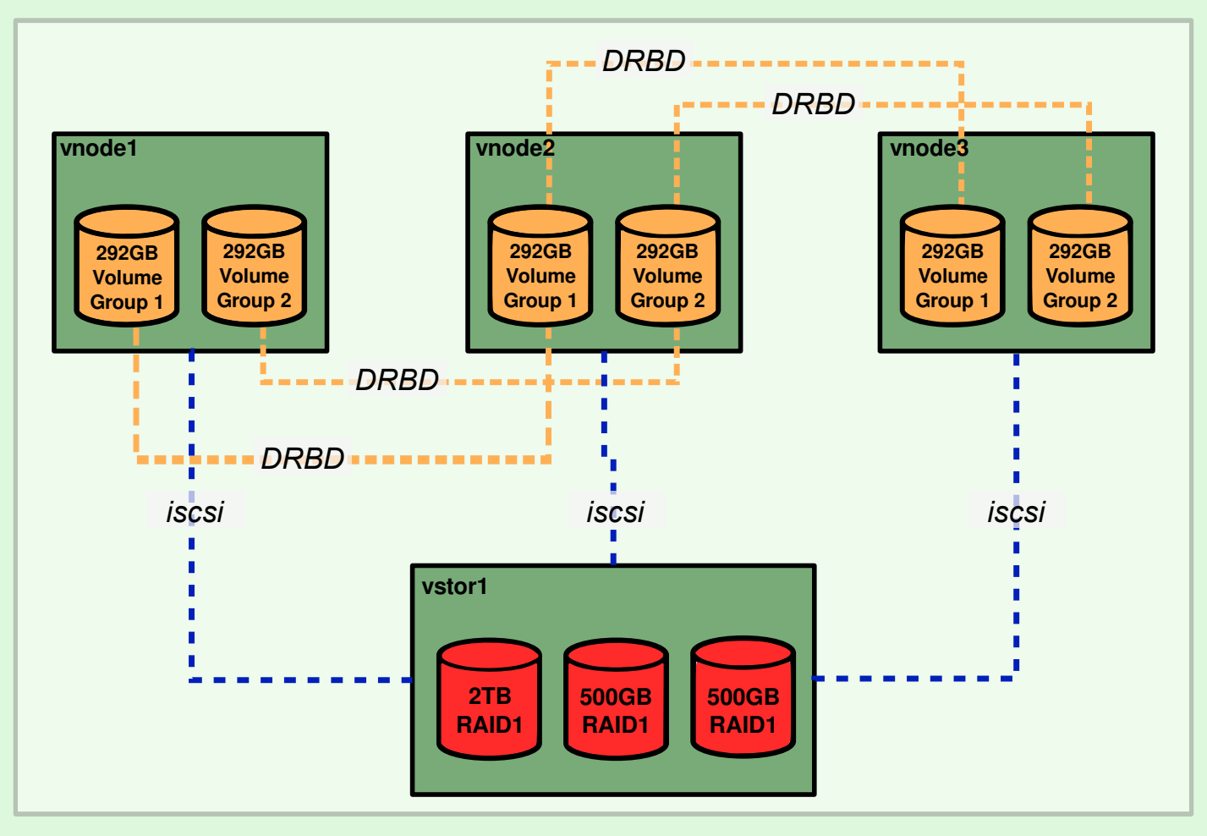
We setted up three vlans for the cluster.

- the network vlan: this is the server network, shared by all the virtual instances ;
- the DRBD vlan: dedicated to DRBD synchronization;
- the iSCSI vlan: dedicated to iSCSI devices access;

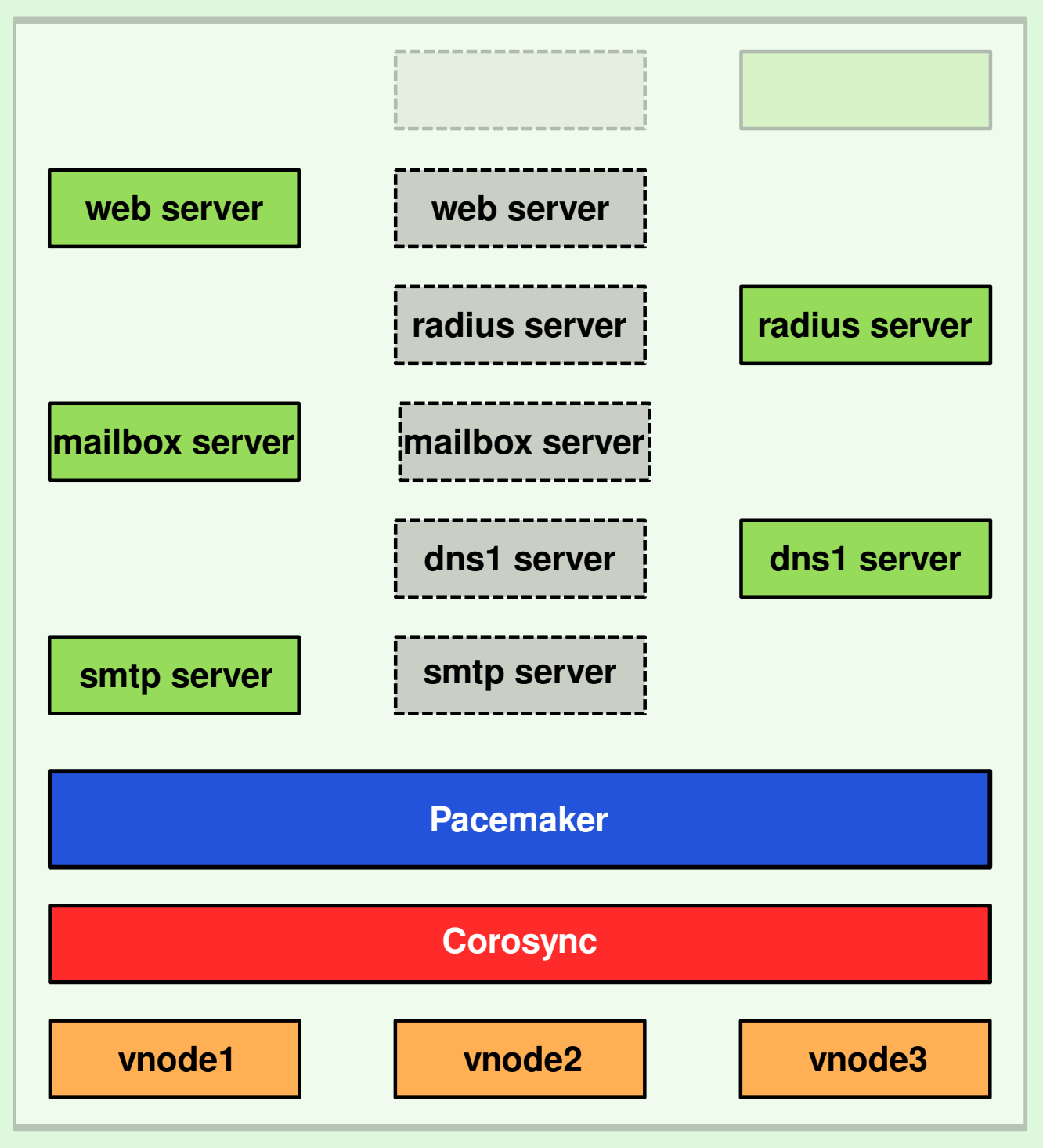


Cluster Storage

On each vnode we created two main LVM volume groups where to put logical volume for virtual instances. On vnode2 we replicated every logical volumes that belong to HA virtual instances.
The node vstor1 offers iSCSI targets that each node can subscribe to.



Cluster Logical Layout



The cluster communication layer is provided by Corosync (the red box). On top of it, runs Pacemaker (the blue box), that is the cluster controller.
The cluster embraces the N+1 model: active virtual instances (green boxes) run on vnode1 and vnode3, ready-to-deploy virtual instances are configured on vnode2 (gray boxes).

Clustering basics

Fail-over
The capability to relocate resources on redundant system in the event of a failure.

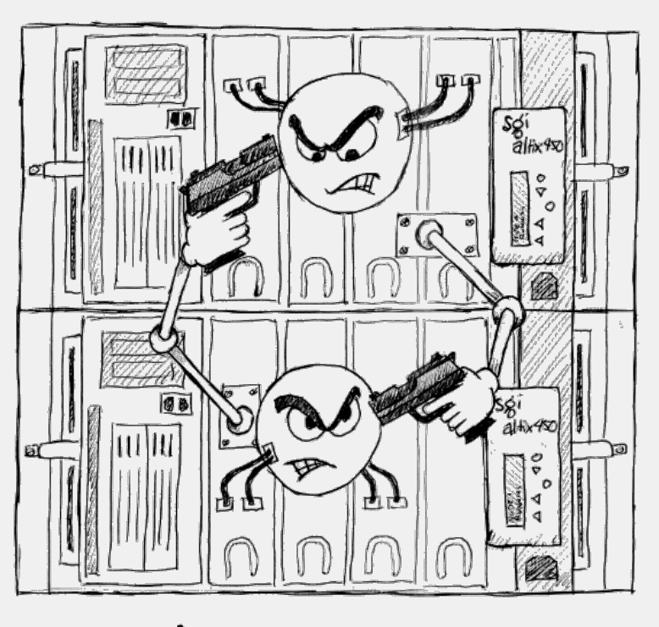
Fail-back
Once a failure is recovered, migrated resources can go back to the system where they belong. Wherever automatic fail-over is the espected behaviour of a cluster engine in case of failures, automatic fail-back is often deemed dangerous.

Simmetry
In a simmetric cluster each node can run every resource. On the contrary on an asimmetric cluster, each resource should be configured to run on a specific node.

Stickiness
Stickiness is a resource property that determine its tendency to run on a specific node. It is extremely important to determine the behaviour of a resource after a fail-over event: if it is 0 fail-back can occur, if it is INFINITY, fail-back is inhibited.

Split-brain
When a cluster node mistakenly believes that all the other node are down and it attempt to start another instance of a running resource, we have a split-brain situation.

STONITH
STONITH --- Shoot The Other Node In The Head --- is a set of mechanisms (fencing) used to prevent split-brain situation.



DON'T ANYBODY MOVE...

Configurations

DRBD

```
common {
  syncer { rate 100M; }
}

# VM sentinel
resource sentinel-root {
  protocol C;
  on vnode1 {
        device  /dev/drbd1;
        disk    /dev/vg2/sentinel-root;
        address 192.168.0.1:7789;
        meta-disk       internal;
        }
  on vnode2 {
        device  /dev/drbd1;
        disk    /dev/vg2/sentinel-root;
        address 192.168.0.2:7789;
        meta-disk       internal;
        }
}
[...]
```

High available virtual instances uses drbd devices.
Each drbd devices is composed of two logical volumes located on different physical nodes.
Since we are free to decide where to locate each logical volume, it is possible to use logical volumes belonging to different volume group to optimize disk utilization.

Xen

```
kernel      = '/boot/vmlinuz-2.6.26-2-xen-amd64'
ramdisk     = '/boot/initrd.img-2.6.26-2-xen-amd64'
memory      = '1024'
root        = '/dev/xvda2 ro'
disk        = [
                  'drbd:sentinel-swap,xvda1,w',
                  'drbd:sentinel-root,xvda2,w',
              ]
name        = 'sentinel'
vif         = [ 'mac=00:16:3E:61:01:03' ]
on_poweroff = 'destroy'
on_shutdown = 'destroy'
on_reboot   = 'destroy'
on_crash    = 'destroy'
extra     = 'clocksource=jiffies'
```

Virtual instances are plain old xen virtual machine. The only relevant change made to the standard Xen configuration is the disabling of Xendomain, since we want to manage the start and stop of virtual instances with the cluster controller.
Xen directly supports the use of drbd devices to attain high availability. Xen is capable of set the drbd device to primary on demand, and it also prevent split brain refusing to start an instance if the drbd device is already primary on another physical node.

Corosync

```
[...]
  interface {
            ringnumber: 0
            bindnetaddr: 192.168.0.0
            mcastaddr: 239.255.1.1
            mcastport: 5405
  }

  interface {
            ringnumber: 1
            bindnetaddr: 131.114.28.0
            mcastaddr: 239.255.2.1
            mcastport: 5405
  }
[...]
```

Once you have configured the interface, or the interfaces, on which Corosync will communicate, the daemon works out of the box.
In our case, we choosed to configure two interfaces for redundancy.

Pacemaker

```
$ crm configure show
node vnode1 \
 attributes standby="off"
node vnode2 \
 attributes standby="off"
node vnode3 \
 attributes standby="off"
[...]
primitive sentinel ocf:heartbeat:Xen \
 params xmfile="/etc/xen/ha/sentinel.cfg" shutdown_timeout="5" \
 op monitor interval="10s" \
 op start interval="0s" timeout="60" \
 op stop interval="0s" timeout="40s" \
 meta target-role="Started"
[...]
property $id="cib-bootstrap-options" \
 dc-version="1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b" \
 cluster-infrastructure="openais" \
 expected-quorum-votes="3" \
 stonith-enabled="false" \
 last-lrm-refresh="1285163787" \
 symmetric-cluster="false"
rsc_defaults $id="rsc-options" \
 resource-stickiness="INFINITY"
```

Pacemaker provides a simple console --- the cluster resource manager, aka **crm** --- for all the cluster operations: from configuration to resources start/stop operations, from node management to resources relocation.
The cluster configuration is stored in XML, but you don't have to deal with it, since every configuration directive is setted via the console, and presented as a simple properties file.
Virtual instances are simple resources and this makes all the architecture scalable and readable.

The Benefits

The new serverfarm reached all the desired targets:

hardware usage consolidation: we have four main nodes running twenty virtualized instances;

decoupling server deployment: we can setup a new instances in minutes using predefined OS images;

simplified servers management: we can manage and monitor each server from one coherent console;

Ubiquitous management for free

Pacemaker configuration and management console is powerful and simple, but best of all it does not even require a server, since each node runs the cluster engine management daemon.

A custom solution with a standard soul

The main benefit of this solution is that, even if every component is fully customizable, it is still standard:

- DRBD volumes are builded on custom logical volumes, but are fully manageable with dbrd tools;

- Xen virtual machine are plain old Xen instances, and are still managed by xen xools even after they are configured as cluster resources;

No lock-in and no constraints

Compared to other virtualization frameworks, or complete solutions like Ganeti, the cluster based on Corosync/Pacemaker puts little or no constraints on resources definition and management. Moreover, since resources does not have to be modified to be manageable by the cluster, you are miles away from lock-in scenarios.

The new serverfarm...

Cloud Computing: Thanks, But No Thanks

Cloud Computing Solutions are becoming more and more familiar in the academia. The on demand delivery of computing resources paradigma seem to really fit the needs of a community of users well separated from the system administrators. It is not difficult to find scenarios where such a computing model is helpful and valuable. The question is if it is also a valid answer for all the centers that provide computing resources.

Is all the computing "on demand"?

In data centers like our, the "demand" of computing resources is, or should be, well-known since the beginning, and the final users of the computing platforms --- as distinct from the users of the services provided by that computing platform --- are also the computing resources admins.

Is Cloud Computing helpful even without the "on demand"?

Couldn't it be that Cloud Computing solutions are better computing resources management platforms even if you don't really need the "on demand" part? Well, at least the resources that should be allocated to the cloud computing platform to manage the "on demand" layer are wasted. Anyway, it could be that for big organizations the overhead of the management resources is relatively small. But in data centers like our, where it is not uncommon to have just few physical nodes, those resources could be better used.

What about constraints?

The smooth deployment model of Cloud Computing could have a backside: virtual instances are usually heavely constrained in terms of storage assignment, hardware access and network management.

Conclusions

It worth it

The Corosync/Pacemaker cluster requires some work by the system administrator to be setted up, and the standard cluster configuration is far from being ready for deployment. It is also true that it could require some time to be fully understood. But once you have a good grasp of all the bits and pieces, you will appreciate the polished structure of your cluster.

Open standards and why they matter

The Corosync Cluster Engine is an open source derivative work of the Service Availability Forum, and it is compliant with the APIs defined by the forum. Pacemaker, the cluster resource manager, is an open source project backed by RedHat and Novell, and it has a strong legacy: the Heartbeat standard. That means that every piece of software that is compliant with those standards will work on your cluster, as for example the DRBD Management Console developed by LINBIT.