VirtuaLinux
Storage Abstraction Layer for Efficient Virtual Clustering

http://sourceforge.net/projects/virtualinux/

Marco Aldinucci, Massimo Torquati, Marco Vanneschi
Computer Science Dept., University of Pisa, Italy

Pierfrancesco Zuccato
Eurotech Italy
Outline

- VirtuaLinux basics
  - motivations
  - which problems VirtuaLinux cope with
  - architecture: big picture

- VirtuaLinux features
  - high-availability
    - masterless cluster
    - diskless cluster
    - storage virtualization
  - consolidation
    - virtual cluster management tools
  - develop for cluster without a cluster
    - multi tier distribution

- Experiments & conclusions
VirtuaLinux aims

- **Clusters**
  - a collection of homogenous but independent machines
  - are fragile
    - master node is a single point of failure
    - disks are a common source of failure
  - complex to install and maintain
    - a proper installation and configuration may require days
    - skilled administrators are required
      - root account power is a common source of misconfiguration during production
  - they are shared machines
    - a single configuration does not match user expectations
      - ... I need CentOS, I prefer Ubuntu, I believe in Windows ...

- VirtuaLinux aims to attack these problems
  - Not surprisingly, the project has been founded by an HW producer
Virtualization: a brand new idea ...

Christopher Strachey published a paper titled *Time Sharing in Large Fast Computers* in the International Conference on Information Processing at UNESCO, New York, in June, 1959. Later on, in 1974, he clarified in an email to Donald Knuth that:

"... [my paper] was mainly about multi-programming (to avoid waiting for peripherals) although it did envisage this going on at the same time as a programmer who was debugging his program at a console. I did not envisage the sort of console system which is now so confusingly called time sharing.". Strachey admits, however, that "time sharing" as a phrase was very much in the air in the year 1960.

Robert P. Goldberg describes the then state of things in his 1974 paper titled *Survey of Virtual Machines Research*. He says: "Virtual machine systems were originally developed to correct some of the shortcomings of the typical third generation architectures and multi-programming operating systems - e.g., OS/360."

Anyway, it works (quite often)

- at the bottom line it is a well known tool: abstraction
  - high-level (e.g. JVM); medium-level (e.g. FreeBDS jails); low-level (Simulazione [e.g. Cell], Binary translation [e.g. VMware, Qemu, ...], paravirtualization [Xen, KVM, ...])

makes it possible

- consolidate different OSes in a single HW
- share HW and SW resources
- insulate classes of users and resources
Cluster: a quite classic configuration

- Diskless blades + external storage (SAN/NAS)
  - Fiber/Infiniband SAN-RAID are fast and robust
    - they are enforced at HW level, irrespectively of the OS
    - sometime enforced by law (e.g. USA’s Sarbenes-Oxley)
  - Any existing Linux distribution for this configuration?
    - A plethora of them, but ...
    - they are not standard distributions
      - typically services and their paths require substantial re-configuration
      - complex, require specialized initrd
      - SO update not easy (cannot rely on standard update tools)
VirtuaLinux approach

- A meta-distribution, conceptually (⇒ standardization)
  - choose a Linux distribution and then configure it for clusters
    - Ubuntu, Debian, CentOS, ...
  - the guest OS is not modified, just properly configured

- Master-less (⇒ robustness)
  - no master node (all nodes cooperatively behave as master node)

- Disk-less (⇒ robustness, flexibility)
  - each physical node access to a private and a cluster shared volume
    - volumes on the iSCSI-attached SAN are virtualized by way of VirtuaLinux storage virtualization layer

- Transparently supports Virtual Clusters (VC) (⇒flexibility)
  - tools for VC deployment, mapping, lifecycle control, etc.
  - currently based on Xen paravirtualization
  - VCs are insulated one each other, just share physical resources
Virtual Clusters (VC)

- Natural evolution of VM idea to cluster level
- a collection of coordinated virtual nodes
  - each one being a VM with its own
    - Vcpu
    - Vstorage: private and VC-wide shared
    - virtual networking: VC-private and inter VCs
    - VM technology neutral
      - several options are possible: VMware, Xen, QEMU, ....
      - independent from VM technology
      - quality of VC improves while quality of VM improves
Virtual Clusters (VC)

Physical Cluster + external SAN
InfiniBand + Ethernet
4 Nodes x 4 CPUs
Cluster InfiniBand 192.0.0.0/24
Cluster Ethernet 192.0.1.0/24
Internet Gateway 131.1.7.6
Virtual Clusters (VC)

Virtual Cluster "green"
4VMs x 1VCPUs
10.0.3.0/24

Physical Cluster + external SAN
InfiniBand + Ethernet
4 Nodes x 4 CPUs
Cluster InfiniBand 192.0.0.0/24
Cluster Ethernet 192.0.1.0/24
Internet Gateway 131.1.7.6
Virtual Clusters (VC)

Virtual Cluster "tan"
2VMs x 2VCPUs
10.0.1.0/24

Virtual Cluster "green"
4VMs x 1VCPUs
10.0.3.0/24

Physical Cluster + external SAN
InfiniBand + Ethernet
4 Nodes x 4 CPUs
Cluster InfiniBand 192.0.0.0/24
Cluster Ethernet 192.0.1.0/24
Internet Gateway 131.1.7.6
Virtual Clusters (VC)

Virtual Cluster "pink"
4VMs x 4VCPU
10.0.0.0/24

Virtual Cluster "tan"
2VMs x 2VCPU
10.0.1.0/24

Virtual Cluster "green"
4VMs x 1VCPU
10.0.3.0/24

Physical Cluster + external SAN
InfiniBand + Ethernet
4 Nodes x 4 CPUs
Cluster InfiniBand 192.0.0.0/24
Cluster Ethernet 192.0.1.0/24
Internet Gateway 131.1.7.6
Virtual Clusters (VC)

**Virtual Cluster "pink"
4VMs x 4VCPUs
10.0.0.0/24**

**Virtual Cluster "green"
4VMs x 1VCPUs
10.0.3.0/24**

**Physical Cluster + external SAN
InfiniBand + Ethernet
4 Nodes x 4 CPUs
Cluster InfiniBand 192.0.0.0/24
Cluster Ethernet 192.0.1.0/24
Internet Gateway 131.1.7.6**
Virtual Clusters (VC)

Virtual Cluster "tan"
2 VMs x 2 VCPUs
10.0.1.0/24

Virtual Cluster "pink"
4 VMs x 4 VCPUs
10.0.0.0/24

Virtual Cluster "green"
4 VMs x 1 VCPU
10.0.3.0/24

Physical Cluster + external SAN
InfiniBand + Ethernet
4 Nodes x 4 CPUs
Cluster InfiniBand 192.0.0.0/24
Cluster Ethernet 192.0.1.0/24
Internet Gateway 131.1.7.6
Big Picture

diskless SMP blade N1

diskless SMP blade N2

diskless SMP blade Nm

iSCSI SAN - set of RAID disks (e.g. Linux openfiler)

actively or passively replicated OS services S1, S2, ... (e.g. TFTP, DHCP, NTP, LDAP, DNS, ...)

EVMS volumes (disk abstraction layer)

Xen hypervisor (VMM) & Linux kernel

Virtual Cluster "red" (e.g. 2 VMs x blade, 1 CPU per VM)

Virtual Cluster "tan" (e.g. 1 VM x blade, 4 CPUs per VM)

physical cluster services (replicated, HA layer)

virtualized storage for the physical cluster

Infiniband 10x switch

Giga Ethernet switch

VC “red”

VC “tan”
High Availability
by way of active and passive replication
High availability

- **24/7 cluster availability**
  - to be not confused with application-level fault tolerance ... here we would like to ensure that the cluster survive, not its applications

- **High-availability means redundancy**
  - robust hardware
    - e.g. 5 power supplies, 4 independent network switches, ...  
    - iSCSI-over-Infiniband and Fiber channels to storage  
    - RAID storage
  - service replication
    - all nodes are identical, i.e. no master node  
    - all essential services are replicated on all nodes  
    - each node can be hot-swapped, switched on/off with no impact on cluster availability and stability
# How to replicate services (sample)

<table>
<thead>
<tr>
<th>Service</th>
<th>FT model</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DHCP</td>
<td>active</td>
<td>Pre-defined map between IP and MAC</td>
</tr>
<tr>
<td>TFTP</td>
<td>active</td>
<td>All copies provide the same image</td>
</tr>
<tr>
<td>NTP</td>
<td>active</td>
<td>Pre-defined external NTPD fallback via GW</td>
</tr>
<tr>
<td>IB manager</td>
<td>active</td>
<td>Stateless service</td>
</tr>
<tr>
<td>DNS</td>
<td>active</td>
<td>Cache-only</td>
</tr>
<tr>
<td>LDAP</td>
<td>service-specific</td>
<td>Service-specific master redundancy</td>
</tr>
<tr>
<td>IP GW</td>
<td>passive</td>
<td>Heartbeat with IP takeover (via IP aliasing)</td>
</tr>
<tr>
<td>Mail</td>
<td>node-oriented</td>
<td>Local node and relays via DNS</td>
</tr>
<tr>
<td>SSH/SCP</td>
<td>node-oriented</td>
<td>Pre-defined keys</td>
</tr>
<tr>
<td>NFS</td>
<td>node-oriented</td>
<td>Pre-defined configuration</td>
</tr>
<tr>
<td>SMB/CIFS</td>
<td>node-oriented</td>
<td>Pre-defined configuration</td>
</tr>
</tbody>
</table>
A novel boot sequence to support master-less clusters
Install without a master

- How to install a cluster without a master?
  - the process should begin somehow
- Our solution: metamaster
  - one of the node behave transiently as a master, start the process, then become a standard node
- At the end of the installation all nodes are identical
Install without a master

- How to install a cluster without a master?
  - the process should begin somehow
- Our solution: metamaster
  - one of the nodes behaves transiently as a master, start the process, then become a standard node
- At the end of the installation all nodes are identical
How to install a cluster without a master?

- the process should begin somehow
- Our solution: metamaster
- one of the nodes behaves transiently as a master, start the process, then become a standard node
- At the end of the installation, all nodes are identical
How to install a cluster without a master?

- the process should begin somehow

- Our solution: *metamaster*
  - one of the nodes behaves transiently as a master, start the process, then become a standard node

- At the end of the installation, all nodes are identical

Install without a master
Storage virtualization
an efficient, constant time-space solution
for physical and virtual clusters
Single copy OR full replication?
Single copy OR full replication?

Node 1
Phy or Virt

disk or partition 1

dev
boot
etc
lib
usr
tmp
var
...

Node 2
Phy or Virt

disk or partition 2

dev
boot
etc
lib
usr
tmp
var
...

Node n
Phy or Virt

disk or partition n

dev
boot
etc
lib
usr
tmp
var
...

Node 1
Phy or Virt

disk or partition

var -> var_node1
etc -> etc_node1
...

Node 2
Phy or Virt

disk or partition

var -> var_node2
etc -> etc_node2
...

Node n
Phy or Virt

disk or partition

var -> var_node3
etc -> etc_node3
...
Single copy OR full replication?

- Each node (physical or virtual) has its own copy of the whole disk
- Transparent, easy to build and update
- OS does not need customization
- Inefficient in time and space - O(n*size). Identical OS files are replicated

- Each node (physical or virtual) share a disk (a file system, actually)
- Not transparent, complex to build and update
- OS does need customization
- Efficient in time and space - O(size). OS files are not replicated
Physical and virtual storage wish list

- Transparent, flexible, efficient (time and space)
- independent from the running OS
- creation/destruction of volumes should be dynamic
- trivial solution not suitable
  - e.g. 50 nodes x 10 GB x 100MB/s = ~ 2 hours (optimistic forecast)
  - destroy system stability during operation due to high I/O pressure
  - e.g. 50 nodes x 5 GB = 100 GB just for OS files
Peculiarities of VC storage

- The nodes of a VC are homogenous (same OS)
  - 99% of OS-related files are identical in all VMs
  - no reason to have heterogeneous nodes in VC, since we can have many heterogeneous VCs

- Keep these files in single copy
  - the solution, to be transparent, should not exhibit this to the nodes (both physical and virtual)
  - can be done exploiting snapshots

* can be hardly used “as is”
Understanding snapshots
Snapshots

- Usually used for online backups
  - both original and snapshot can be written
  - FS-neutral, transparent
- Supported in standard tools (e.g. LVM, EVMS)
  - in Linux implemented via dm_snapshot module (device mapper)
  - used also in other systems, such as WinXP system recovery machinery
- Can be implemented in several ways
  - copy-on-write, redirect-on-write, split-mirror, ...
- They have been used also to store/share VM images
  - for a single machine, not for clusters ...
Copy-on-write
Copy-on-write

writing on the original
Copy-on-write

writing on the original

writing on the snapshot

Original

Snapshot
Snapshots are not designed for parallel systems

- if used with standard semantics all snapshots should remain active in all nodes (even those not accessing them)
- they are buffered in kernel space, thus consume kernel memory
  - space linear in the number of snapshots system-wide

VirtuaLinux introduces and uses a novel semantics

- relax the standard semantics maintaining the correctness
  - mark as read-only the blocks that will not change in the original (e.g. OS files)
  - enable the deactivation of not used snapshots
    - correct provided the original is read-only
    - standard semantics cannot enforce it since has no way to mark as read-only
  - implemented as EVMS plugins, no kernel space changes
VirtuaLinux storage virtualization
(physical cluster)

**EVMS**
- **volumes**
  - /dev/evms/default
  - /dev/evms/node1
  - /dev/evms/node2
  - ... /dev/evms/noden
  - /dev/evms/shared
- **snapshots**
  - snap_1
  - snap_2
  - ... snap_n
- **regions**
  - R_def
  - Ra_1
  - Ra_2
  - ... Ra_n
  - Rb_1
  - Rb_2
  - ... Rb_n
  - R_shared
- **containers**
  - container_0
- **segments**
  - segm1
  - segm2
  - segm3
- **disks**
  - sda1
  - sda2
  - sda3
  - sdb
  - sdb

**SAN**
VirtuaLinux storage virtualization
(virtual clusters)
Experiments

Vcpu, Vnetwork and Vstorage overhead
**Communication Latency**

<table>
<thead>
<tr>
<th>Benchmark Name</th>
<th>Latency (usec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dom0_IB</td>
<td></td>
</tr>
<tr>
<td>Dom0_IPoIB</td>
<td></td>
</tr>
<tr>
<td>Dom0_Geth</td>
<td></td>
</tr>
<tr>
<td>DomU_IPoIB</td>
<td></td>
</tr>
</tbody>
</table>

---

**Dom0_IB**    Ubuntu Dom0, Infiniband user-space verbs (MPI-gen2)
**Dom0_IPoIB**  Ubuntu Dom0, Infiniband IPoverIB (MPI-TCP)
**Dom0_Geth**   Ubuntu Dom0, Giga-Ethernet (MPI-TCP)
**DomU_IPoIB**  Ubuntu DomU, virtual net on top of Infiniband IPoverIB (MPI-TCP)
Communication Bandwidth

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dom0_IB</td>
<td>Ubuntu Dom0, Infiniband user-space verbs (MPI-gen2)</td>
</tr>
<tr>
<td>Dom0_IPoIB</td>
<td>Ubuntu Dom0, Infiniband IPoverIB (MPI-TCP)</td>
</tr>
<tr>
<td>Dom0_Geth</td>
<td>Ubuntu Dom0, Giga-Ethernet (MPI-TCP)</td>
</tr>
<tr>
<td>DomU_IPoIB</td>
<td>Ubuntu DomU, virtual net on top of Infiniband IPoverIB (MPI-TCP)</td>
</tr>
</tbody>
</table>
# Performance (CPU & SO)

<table>
<thead>
<tr>
<th>Micro-benchmark</th>
<th>Ub-Dom0 vs CentOS</th>
<th>Ub-DomU vs CentOS</th>
<th>Ub-DomU vs Ub-Dom0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple syscall</td>
<td>+667%</td>
<td>+726%</td>
<td>+7%</td>
</tr>
<tr>
<td>Simple open/close</td>
<td>+36%</td>
<td>+34%</td>
<td>-2%</td>
</tr>
<tr>
<td>Select on 500 tcp fd’s</td>
<td>+51%</td>
<td>+51%</td>
<td>0%</td>
</tr>
<tr>
<td>Signal handler overhead</td>
<td>+112%</td>
<td>+127%</td>
<td>+7%</td>
</tr>
<tr>
<td>Protection fault</td>
<td>+246%</td>
<td>+293%</td>
<td>+13%</td>
</tr>
<tr>
<td>Pipe latency</td>
<td>+115%</td>
<td>+31%</td>
<td>-40%</td>
</tr>
<tr>
<td>Process fork+execve</td>
<td>+143%</td>
<td>+119%</td>
<td>-10%</td>
</tr>
<tr>
<td>float mul</td>
<td>~0%</td>
<td>~0%</td>
<td>~0%</td>
</tr>
<tr>
<td>float div</td>
<td>~0%</td>
<td>~0%</td>
<td>~0%</td>
</tr>
<tr>
<td>double mul</td>
<td>~0%</td>
<td>~0%</td>
<td>~0%</td>
</tr>
<tr>
<td>double div</td>
<td>~0%</td>
<td>~0%</td>
<td>~0%</td>
</tr>
<tr>
<td>RPC/udp latency localhost</td>
<td>+35%</td>
<td>-7%</td>
<td>-31%</td>
</tr>
<tr>
<td>RPC/tcp latency localhost</td>
<td>+35%</td>
<td>-5%</td>
<td>-30%</td>
</tr>
<tr>
<td>TCP/IP conn. to localhost</td>
<td>+32%</td>
<td>+3%</td>
<td>-22%</td>
</tr>
<tr>
<td>Pipe bandwidth</td>
<td>-38%</td>
<td>+51%</td>
<td>+144%</td>
</tr>
</tbody>
</table>
## Virtual Storage Performance

<table>
<thead>
<tr>
<th>Additional layer on top of iSCSI</th>
<th>read</th>
<th>write</th>
<th>rewrite</th>
</tr>
</thead>
<tbody>
<tr>
<td>none (reference raw iSCSI access)</td>
<td>60</td>
<td>88</td>
<td>30</td>
</tr>
<tr>
<td>EVMS standard volume</td>
<td>66</td>
<td>89</td>
<td>32</td>
</tr>
<tr>
<td>EVMS snap, fresh files</td>
<td>63</td>
<td>88</td>
<td>31</td>
</tr>
<tr>
<td>EVMS snap, files existing on original</td>
<td>63</td>
<td>7</td>
<td>31</td>
</tr>
</tbody>
</table>
VirtuaLinux 1.1 (multi tier)

- Based on Ubuntu, kernel 2.6.19-4 (gutsy)
- Designed to support our developers working at home
  - did you have at home a cluster with a SAN?
  - the cluster is simulated by yet another level of virtualization (binary translator, e.g. VMware)
- Three-tiers (two of them virtualized)
  - tier 0, standard linux - simulates the SAN (iscsi-target)
  - tier 1, macchine VMware - simulates the physical cluster
  - tier 2, macchine Xen - nodes of virtual clusters
- Slow, but still, it makes the development possible
  - can be used for demo of parallel apps in conferences
    - if you have at least a 64bit core2duo Intel (≥ Merom) laptop (I haven’t, sorry)
VirtuaLinux many-tier

tier 0
laptop standard linux
+ iSCSI target
VirtualLinux many-tier

tier 0
laptop standard linux
+ iSCSI target

tier 1
virtual cluster
of VM diskless
(VMware or QEMU)
**VirtuaLinux many-tier**

- **tier 1**: virtual cluster of VM diskless (VMware or QEMU)
- **tier 0**: laptop standard Linux + iSCSI target
VirtuaLinux many-tier

**tier 1**
- vEth share
- tier 1 vNet

**tier 2**
- vNet
- VirtuaLinux with VC (Xen)

**tier 0**
- laptop standard linux + iSCSI target
- tier 1 virtual cluster of VM diskless (VMware o QEMU)
- iSCSI+EVMS via tier 1 vNet
Conclusion

❖ Focuses on HPC cluster for industry needs
   ✦ reduce install and maintenance costs
   ✦ makes it possible the consolidation and sharing
   ✦ prevent the destruction of installation at deployment site due to weird administrator actions

❖ Some scientific results
   ✦ some advance in storage virtualization
     ✧ comparable VMware Lab Manager (not opensource)
     ✧ performance (sometime) better than non-virtualized storage

❖ Some industrial results
   ✦ currently deployed on shipped Eurotech HPC clusters
     ✧ subsets of the whole system
   ✦ Graphic version of VC management tools not opensource
We would like to acknowledge Eurotech S.p.A. Italy and University of Pisa for the financial support and hardware

and

all people who contributed to the development, and in particular aldinuc, calillo, gervystar, gobex, massimot, monica_d, patton73, pierfrancesco, spinatel

Virtualinux is opensource under GPL and it is meant to be a continually evolving experimentation framework, thus Please do not hesitate to contact me if you would like propose new ideas or to participate to develop our own

email: aldinuc@di.unipi.it
http://www.di.unipi.it/~aldinuc