VirtuaLinux

An open source HPC clustering solution

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

M. Aldinucci, M. Torquati, et al. (Unipi)
P. Zuccato, A. Gervaso (Eurotech S.p.A.)
Eurotech HPC
http://www.eurotech.it

Pierfrancesco Zuccato
Responsabile di Eurotech HPC
Industry & research

The VirtuaLinux project
- Entirely founded by Eurotech S.p.A.
- Aims to solve industrial problems
  - They do need pure and applied research
- Scientific goals published in A-class conferences
- Developed software released as open source GPL
Problem statement

- **Cluster**
  - collection of (high-density) legacy independent machines connected by means of a LAN
  - are fragile
    - master node is a single point of failure
    - disks-on-blades are a common source of node failure
  - are hard to install and to maintain
    - installation requires days
    - skilled administrator are needed
      - root administrator problem
  - legacy OSes
- High-density blades + external SAN (or NAS)
  - RAID SAN much more efficient & robust
    - Robustness and speed are separately addressed (often at HW level)
    - Sometimes legally required (e.g. USA Sarbenes-Oxley law)
- A number of linux distribution support this configuration
  - They require the customization of the standard distribution
    - typically path and configuration of services

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
A single cluster configuration does not match all user expectation

I want CentOS, I need Ubuntu, I believe in Windows

the last one being much more a religion than an OS

Cluster life-cycle in 5 easy steps

1. the cluster is installed with factory distribution
2. the cluster falls in the hands of site system administrators
3. they mix-up user requirements
   - and they believe to be wizards, in reality they sometimes are sorcerer's apprentices
   - of course wizard exists, but they want to be paid, and this is forbidden by cluster owner religion
4. after two days they destroy the cluster
5. ask for the factory assistance, goto 1
Virtualization: a brand-new solution?

Anyway, it works (it is abstraction, at the end)

- High-level (e.g. JVM)
- medium-level (e.g. FreeBDS jails)
- low-level
  - Simulation (e.g. Cell), Binary translation (e.g. VMware, QEMU, ...), paravirtualization (XEN, KVM, ...)

Make it possible to

- consolidate several OSes
- make it possible to share a physical resource
- insulate user OSes from lower-level OS

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."

[By the way, the term 'virtual machine' was coined in 1968 by Kent Melton at Stanford University. Melton was working on a project called MULTICS, which stood for Multi-Level Time-sharing Information and Computing System. MULTICS was an experimental operating system that was designed to support multiple users running interactive processes on a single computer. The concept of virtual machines was an integral part of MULTICS, allowing different users to run their own processes on the same physical hardware. Melton's work on virtual machines laid the groundwork for future developments in virtualization, including the modern virtual machines used in cloud computing and server virtualization.]

Strachey admits, however, that "time sharing" as a phrase was very much in the air in the year 1960.
VirtuaLinux approach

- A meta-distribution
  - get a Linux distribution and automatically configure it
- Master-less
  - Any master node; nodes are fully symmetrical
- Disk-less
  - Provide the cluster with fully independent virtual volumes starting from a network-attached SAN
  - No customization of the OS are required
- Transparently supporting Virtual Clusters
  - and the tools to manage them
Virtual clusters

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

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

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

Restart & Remap tan

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)
Each virtual node (VM) of a VC is a virtual machine that can be configured at creation time. It exploits a cluster-wide shared storage.
Each VC exploits a private network and can access the cluster external gateway.
VMs of a VC can be flexibly mapped onto the cluster nodes.
VCs can be dynamically created, destroyed, suspended on disk.

Disks Abstraction Layer
A set of private and shared EVMS volumes are mounted via iSCSI in each node of the cluster:
A private disk (/root) and an OCFS2/GFS cluster-wide shared SAN are mounted in each node.
EVMS snapshot technique is used for a time and space efficient creation of the private remote disk.
A novel plug-in of EVMS has been designed to implement this feature.

Linux OS services
All standard Linux services are made fault-tolerant via either active or passive replication:
Active: Services are started in all nodes; a suitable configuration enforces load balance on client requests. E.g. NTP, DNS, TFTP, DHCP.
Passive (primary-backup): Linux HA with heartbeat is used as fault detector. E.g. LDAP, IP gateway.

Kernel Basic Features
All standard Linux modules.
Xen hypervisor, supporting Linux paravirtualization, and Microsoft Windows via QEMU binary translation (experimental).
Network connectivity, including Infiniband userspace verbs and IP over Infiniband.
iSCSI remote storage access.
OCFS2 and GFS shared file systems.

The Big Picture
High Availability
by way of the replication of services
High availability

- 24/7 cluster availability
- Redundant hardware
  - 5 Power supplies, 4 independent network switches, ...
  - iSCSI-over-Infiniband or Fibre channel attached RAID
- Redundant services
  - All nodes are identical, and there is no master
  - All OS services are replicated on all nodes
    - Any blade can be detached, or crash at any point in time with no impact on system availability
Provide services without a (single) server

<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>
Novel boot sequence

supporting the boot of master-less systems
How to install a masterless cluster?

- Chicken and egg problem!

- **Solution: Metamaster**
  - A transient master node that is then transformed in a standard node
  - At the end of the installation, all nodes are identical and provide redundant services.
  - No master node, fully symmetrical nodes
Storage Virtualization

a transparent constant time-space solution
Full vs no replication

- Each node (physical or virtual) has its own copy of the whole disk
- Transparent, easy to build and update
- OS does not need customisation
- Inefficient in time and space - $O(n \times \text{size})$. 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 customisation
- Efficient in time and space - $O(\text{size})$. OS files are not replicated
VC requirements

- Needs both transparency and time-space efficiency
  - Should be independent from the OS
  - “Frequent” & very time consuming operation
    - e.g. 50 nodes x 10 GB x 100MB/s = ~ 2 hour
      - very optimistic forecast
      - just to start/suspend the VC (and the system become irresponsive)
  - Consumes lot of disk space
    - e.g. 50 nodes x 5 GB = 100 GB
      - for each VC and for the OS only - no user data here
VC storage properties

- Nodes of a VC are homogenous (same OS)
  - 99% of OS-related files are identical in all VMs
  - No reason to have heterogeneous VCs, just start more VCs of different kind
  - It is low-level virtualization
    - the virtualized is similar to the real
- Just keep them in a single copy
  - but don’t tell to the virtual nodes, they believe to be fully independent one each other
  - A novel usage of the snapshotting technique
    - copy-on-write
Snapshot technique

- Time: 0001
- Write: 0010
- Snap t1
- Time: 0011
- Write: 0111
- Snap t2
- Time: 1011
- Write: 1111
- Time: 0010
- Snap t1
- Time: 0010
- Snap t2
- Time: 1111
Snapshots

- Used to build online backups
  - Both original and snapshots can be written
- File system independent, transparent
- Supported by several tools, e.g. EVMS, LVM2
  - Implemented by linux standard kernel
  - dm_snapshot module (device mapper)
- Can be implemented in several ways
  - copy-on-write, redirect-on-write, split-mirror, ...
Copy-on-write

Writing on original

Writing on snapshot

Originale

Snapshot
Concurrent snapshots

- Snapshots not designed for concurrency
  - All snapshot should be kept active in all nodes
  - Linux cannot keep active more than 8 snapshots per node

- Novel semantics for snapshots
  - Relax snapshots semantics while maintaining correctness
  - “Mark” as read-only parts on the blocks
Storage virtualization

file system
ext3 swap ext3 swap ...
/dev/evms/default /dev/evms/swap1 /dev/evms/swap2 ...

snaphots
snap_1 snap_2 ...
snap_n

regions
R_def Ra_1 Rb_1 Ra_2 Rb_2 ...
Ra_n Rb_n R_shared

containers
container_0

segments
segm1 segm2 segm3

disks
sda1 sda2 sda3 sdb

EVMS volumes
/dev/evms/node1 /dev/evms/node2 ...
/dev/evms/node2 /dev/evms/node2 ...
/dev/evms/swap1 /dev/evms/swap2 ...
/dev/evms/swapn

/sda ext3 swap sda2 ext3 swap

cluster ...

SAN
Storage virtualization (VCs)
Bandwidth

<table>
<thead>
<tr>
<th>Bandwidth (MB/s)</th>
<th>Sendrecv size (Bytes) - 2 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>0</td>
<td>64</td>
</tr>
<tr>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>0</td>
<td>1K</td>
</tr>
<tr>
<td>0</td>
<td>4K</td>
</tr>
<tr>
<td>0</td>
<td>16K</td>
</tr>
<tr>
<td>0</td>
<td>64K</td>
</tr>
<tr>
<td>0</td>
<td>256K</td>
</tr>
<tr>
<td>0</td>
<td>1M</td>
</tr>
<tr>
<td>0</td>
<td>4M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bandwidth (MB/s)</th>
<th>Sendrecv size (Bytes) - 4 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>0</td>
<td>64</td>
</tr>
<tr>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>0</td>
<td>1K</td>
</tr>
<tr>
<td>0</td>
<td>4K</td>
</tr>
<tr>
<td>0</td>
<td>16K</td>
</tr>
<tr>
<td>0</td>
<td>64K</td>
</tr>
<tr>
<td>0</td>
<td>256K</td>
</tr>
<tr>
<td>0</td>
<td>1M</td>
</tr>
<tr>
<td>0</td>
<td>4M</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)
## CPU & OS Performances

<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>
### Storage Virtualization Performances

<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>

**Table 1.** Performances (MBytes/s) of the proposed storage abstraction layer. Results are referred to Bonnie block read/write/rewrite benchmarks on a iSCSI-attached SAN.


5. VirtuaLinux at Linux Day 2007 Meeting (invited talk, October 27th, 2007)

6. VirtuaLinux at NSS07 Conference (invited talk, November 27th, 2007,
Xen architecture

- Dom0 VM0
  - Device Manager and Control SW
  - Para-virtualized GuestOS (e.g. Linux)
    - Native Device Drivers
    - Back-End Device Drivers
  - Unmodified User Software

- DomU VM1
  - Unmodified User Software
  - Para-virtualized GuestOS (e.g. Linux)
    - Front-End Device Drivers
  - SMP

- DomU VM2
  - Unmodified User Software
  - Para-virtualized GuestOS (e.g. Linux)
    - Front-End Device Drivers
  - Virtual CPU

- DomU VM3
  - Unmodified User Software
  - Para-virtualized GuestOS (e.g. WinXP)
    - Front-End Device Drivers
  - Virtual MMU

- QEMU bin translator

- Hardware (SMP, MMU, physical memory, Ethernet, SCSI/IDE)
  - Control IF
  - Safe HW IF
  - Event Channel
  - Virtual CPU
  - Virtual MMU

- Hypervisor (Xen Virtual Machine Monitor)