VirtuaLinux: Virtualized High Density Clusters with no Single Point of Failure


Parco 2007
RWTH Aachen, Forschungszentrum Jülich (Germany) 3rd-7th September 2007
OVERVIEW: COMMON FLAWS

- Cluster configuration is hard to evolve as needs change (reinstallation)
- One cluster → one environment (how to cope with GRIDs, Beowulf, etc?)
- Strong coupling between physical and logical architectures
- Single point of failure (master node)
- Multiple points of failure (local hard disks)
- Resource waste (OS replication for homogeneous nodes)
- Configuration is not reusable (installation procedure must be redone for identical machines)
OVERVIEW: VirtuaLinux

Virtual clusters “VC”

Virtual storage layer

Service replication

Standard kernel

... and the tools to manage this
CONTENTS

OVERVIEW

VIRTUALIZATION: CLUSTER OS

VIRTUALIZATION: CLUSTER STORAGE

FAULT TOLERANCE

TOOLS

DOES IT WORK?

CONCLUSIONS
VIRTUALIZATION: CLUSTER OS

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

**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

**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
VIRTUALIZATION: CLUSTER OS

- Three layers: physical, host, guest clusters
- The host cluster layer hides and protects hardware details
- Guest clusters are clusters of Virtual Machines (VM) that are totally insulated (Virtual Clusters, VC)

Key features of VCs:
- Crashes and instabilities are confined
- VCs can have different configurations and OS (Beowulf, Grid, RH, Ubuntu, Windows).
- Guest administrators don’t have to be very skilled (limited damage)
- VC and VMs can be preconfigured and downloaded from repositories
- VMs within a VC can be moved between physical nodes
- Possibly, superscalability (I/O bound and CPU bound VCs running on the same physical nodes)
A physical node running 3 VMs (2 to 3 VCs nodes. One runs Windows through QUEMU). Xen provides the virtualization
OVERVIEW

VIRTUALIZATION: CLUSTER OS

VIRTUALIZATION: CLUSTER STORAGE

FAULT TOLERANCE

TOOLS

DOES IT WORK?

CONCLUSIONS
VIRTUALIZATION: CLUSTER STORAGE

VirtuaLinux, EVMS and iSCSI
Virtualization: Cluster Storage

- Storage virtualization in VirtuaLinux means EVMS+creative use of snapshots on an iSCSI SAN
- iSCSI provides standard, simple, cheap and cluster wide access to the remote, redundant disks, allowing diskless nodes
- iSCSI can reuse existing infrastructures (GigEth, Infiniband). No additional cost, cabling, adapters or drivers
- EVMS insulates the physical details
- EVMS provides a single, unified system for storage management
- EVMS logical volume naming avoids the use of raw device names (iSCSI devices connected to different nodes would have different names)
VIRTUALIZATION: CLUSTER STORAGE

- Generally, all VC virtual nodes share the same OS image, with minor differences.
- 1000 virtual nodes -> 1000 replicas. Disk and time waste.
- The same is true for classic clusters.
- Snapshot is traditionally used to backup volumes, but can be used to create replicas (volumes).
- Modified snapshot technique:
  - Create the volume and install the OS image.
  - Make N snapshots of the original (for N nodes).
  - The snapshots are created using ONLY metadata and pointers to the original copy.
  - Creation time of N snapshots almost independent from N (seconds).
  - Modifications are lightweight: a modification to a file just involves the copy and update of the single file.
- Memory buffers in kernel limit the N. VirtuaLinux extends EVMS semantics (one line of C code) to allow any N.
As for the physical cluster, each VC node has a VC private virtual shared storage (OCFS2 distributed FS), a private virtual disk and a private virtual swap area. Private virtual disks are functionally identical to traditional local disks. Virtual means that the volume is a snapshot.
CONTENTS

- OVERVIEW
- VIRTUALIZATION: CLUSTER OS
- VIRTUALIZATION: CLUSTER STORAGE
- FAULT TOLERANCE
- TOOLS
- DOES IT WORK?
- CONCLUSIONS
Traditionally, clusters have a master node. Great risk!

VirtuaLinux is a masterless architecture

Active and passive replication (primary-backup) of services

Service classification:

- Stateless: Active replication (e.g. IB manager). Clients automatically locate the server (broadcasting, etc). Most reactive server replies
- Stateful: Passive replication (IP Gateway). Only one server exists and is statically known to clients (IP, MAC). Heartbeat with IP takeover between primary and backup servers
- Node oriented: failure of node specific services is potentially catastrophic for the node only (local SSH, NFS). The rest of the cluster is unaffected. Outside the scope of VirtuaLinux. Scheduler level?
- Self-healing: fault tolerance is embedded in the service (NTP)
## FAULT TOLERANCE

<table>
<thead>
<tr>
<th>Service</th>
<th>Fault-tolerance</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 DSN</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 on 2 nodes with IP takeover (HA)</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>
FAULT TOLERANCE

How to install a masterless cluster? Chicken and egg problem!
Solution: Metamaster! A temporary installation node that is then transformed in a standard node
At the end of the installation, all nodes are identical and provide redundant services. Bye bye master node
TOOLS

VirtuaLinux is a set of scripts, independent of the Linux Distribution

VirtuaLinux Virtual Cluster Management (VVCM) is a subset of scripts that allow the creation and management of VCs

Main components are:

- Database of the physical and virtual clusters (includes the mapping between physical and virtual clusters, virtual nodes, etc)
- Command line library for the creation, activation and destruction of a VC (VC_Create, VC_Control, VC_Destroy)
- Communication layer (for staging and executing VMs)
- VC start time support for dynamic configuration of network topology and routing policies of the physical nodes
CONTENTS

- OVERVIEW
- VIRTUALIZATION: CLUSTER OS
- VIRTUALIZATION: CLUSTER STORAGE
- FAULT TOLERANCE
- TOOLS
- DOES IT WORK?
- CONCLUSIONS
### DOES IT WORK?

<table>
<thead>
<tr>
<th>Micro-benchmark</th>
<th>Unit</th>
<th>Ub-Dom0</th>
<th>Ub-DomU</th>
<th>CentOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple syscall</td>
<td>usec</td>
<td>0.6305</td>
<td>0.6789</td>
<td>0.0822</td>
</tr>
<tr>
<td>Simple open/close</td>
<td>usec</td>
<td>5.0326</td>
<td>4.9424</td>
<td>3.7018</td>
</tr>
<tr>
<td>Select on 500 tcp fd’s</td>
<td>usec</td>
<td>37.0604</td>
<td>37.0811</td>
<td>75.5373</td>
</tr>
<tr>
<td>Signal handler overhead</td>
<td>usec</td>
<td>2.5141</td>
<td>2.6822</td>
<td>1.1841</td>
</tr>
<tr>
<td>Protection fault</td>
<td>usec</td>
<td>1.0880</td>
<td>1.2352</td>
<td>0.3145</td>
</tr>
<tr>
<td>Pipe latency</td>
<td>usec</td>
<td>20.5622</td>
<td>12.5365</td>
<td>9.5663</td>
</tr>
<tr>
<td>Process fork+execve</td>
<td>usec</td>
<td>1211.4000</td>
<td>1092.2000</td>
<td>498.6364</td>
</tr>
<tr>
<td>float mul</td>
<td>nsec</td>
<td>1.8400</td>
<td>1.8400</td>
<td>1.8200</td>
</tr>
<tr>
<td>float div</td>
<td>nsec</td>
<td>8.0200</td>
<td>8.0300</td>
<td>9.6100</td>
</tr>
<tr>
<td>double mul</td>
<td>nsec</td>
<td>1.8400</td>
<td>1.8400</td>
<td>1.8300</td>
</tr>
<tr>
<td>double div</td>
<td>nsec</td>
<td>9.8800</td>
<td>9.8800</td>
<td>11.3300</td>
</tr>
<tr>
<td>RPC/udp latency localhost</td>
<td>usec</td>
<td>43.5155</td>
<td>29.9752</td>
<td>32.1614</td>
</tr>
<tr>
<td>RPC/tcp latency localhost</td>
<td>usec</td>
<td>55.0066</td>
<td>38.7324</td>
<td>40.8672</td>
</tr>
<tr>
<td>TCP/IP conn. to localhost</td>
<td>usec</td>
<td>73.7297</td>
<td>57.5417</td>
<td>55.9775</td>
</tr>
<tr>
<td>Pipe bandwidth</td>
<td>MB/s</td>
<td>592.3300</td>
<td>1448.7300</td>
<td>956.21</td>
</tr>
</tbody>
</table>

LM microbenchmark:
4 nodes, dual Opteron 280, 8GB ram, GigEth, SDR Infiniband
Environments: Host (Ub-Dom0), Guest (Ub-DomU), not virtualized (CentOS)
## DOES IT WORK?

<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>

Virtualization impacts communications and system calls, not math.
DOES IT WORK?

Intel MPI Benchmarks (VMAPICH): Bandwidth, collective communications
DOES IT WORK?

Intel MPI Benchmarks (VMAPICH): Latency
(Para)Virtualization impact syincalls heavily, but the real cost for computationally intensive applications is not clear.

(Para)Virtualization has no cost for processor instructions (e.g. math).

(Para)Virtualization significantly penalizes communication bandwidth and latency, but:

- Current native IB drivers (with user space verbs) cannot be used within VMs.
- TCP is used (IPoIB) within VMs. This is a major source of overhead.
- Most overhead will be eliminated when user-space IB verbs for Xen VM are available (currently under development, see J. Liu, W. Huang, B. Abali and D.K. Panda. High Performance VMM-Bypass I/O in Virtual Machines. USENIX Annual Technical Conference 2006, Boston, MA, May 2006.)
CONCLUSIONS

- VirtuaLinux enables a virtualized, diskless, masterless cluster architecture
- Virtual clusters are insulated from the physical and host cluster, allowing multiple simultaneous VCs on the same hardware
- Storage virtualization coupled with a novel snapshot technique dramatically reduces installation time and permits a centralized management of the VCs
- Storage virtualization permits the backup and restore of entire VCs
- The absence of an individual master and local hard disks replaced by a SAN of redundant arrays avoids single points of failure
- IB Verbs within VMs are absolutely required for real use of VirtuaLinux in a production environment, but they are coming...
ACKNOWLEDGMENTS

VirtuaLinux is an ongoing experiment of Eurotech SpA and the HPC laboratory of the Computer Science Department of the University of Pisa.


VirtuaLinux project has been supported by the initiatives of the LITBIO Consortium, founded within FIRB 2003 grant by MIUR, Italy.

We are grateful to Peter Kilpatrick for his help in improving the presentation.
"On résiste à l'invasion des armées; on ne résiste pas à l'invasion des idées."

Victor Hugo, Paris 1877