The VirtuaLinux Storage Abstraction Layer for Efficient Virtual Clustering

Marco Aldinucci, Massimo Torquati, Marco Vanneschi
Computer Science Department
University of Pisa, Italy
{aldinuc, torquati, vannesch}@di.unipi.it

Pierfrancesco Zuccato
Eurotech S.p.A.
Amaro (Udine), Italy
p.zuccato@eurotech.it

Abstract

VirtuaLinux is a meta-distribution that enables a standard Linux distribution to support robust physical and virtualized clusters. VirtuaLinux helps in avoiding the "single point of failure" effect by means of a combination of architectural strategies, including the transparent support for disk-less and master-less cluster configuration. VirtuaLinux supports the creation and management of Virtual Clusters in seamless way: VirtuaLinux Virtual Cluster Manager enables the system administrator to create, save, restore Xen-based Virtual Clusters, and to map and dynamically re-map them onto the nodes of the physical cluster. In this paper we introduce and discuss VirtuaLinux virtualization architecture, features, and tools, and in particular, the novel disk abstraction layer, which permits the fast and space-efficient creation of Virtual Clusters.

1. Introduction

Physical clusters are usually deployed to improve performance and/or availability over that provided by a single computer. A cluster is composed of a network of complete computers (nodes), each of them running *its own copy* of an OS, which can be either a part of a distributed OS (e.g. Single System Image OSes) or a fully standard OS (e.g. Linux). The latter solution permits a finer control on services deployment, and the full reuse of the expertise of administrator, who can leverage on a range of tools for the collective and/or centralized management of the nodes: from simple scripts (e.g. `rdist`) to complete software packages (e.g. Sun Grid Engine [25]).

Frequently, clusters are equipped with an external shared disk (Storage Area Network or SAN), which simplifies the administration of the storage since cables and storage devices do not have to be physically moved to move storage from one server to another. SANs tend to increase storage capacity utilization, since multiple servers can share the same growth reserve, and if compared to disks that a high-density cluster can accommodate, exhibit better performances and reliability since they are realized by arranging high-speed high-quality disks in RAID [17]. SANs also tend to enable more effective disaster recovery processes.

Typically, the nodes of a cluster are homogeneous at the hardware level. However, one of them acts as the master of the cluster, whereas the other nodes depend on it for several services, such as file sharing, user authentication, network routing and resolution, time synchronisation. The master is usually statically determined at the installation time. The master node is a single point of failure for cluster availability since it hosts services that are critical for cluster operations.

Due to their cost, clusters are typically shared resources, and rarely a single configuration or even a single OS can be adapted to supply all applications and users' needs. Classic solutions, like static cluster partitioning and multiple boots, are not flexible enough to consolidate several user environments and require a consistent configuration effort. Since configuration involves distinct OS copies in different nodes, any configuration mistake may seriously impair cluster stability and is difficult to undo.

In this paper we introduce VirtuaLinux, an innovative Linux meta-distribution able to support the installation and the management of a disk-less, master-less cluster with a standard Linux distribution (e.g. CentOS, Ubuntu). VirtuaLinux addresses the mentioned robustness and flexibility problems of standard cluster installations via cluster-level resource virtualization, i.e. Virtual Clusters (VCs). This paper presents VirtuaLinux VC architecture, and in particular, the VirtuaLinux disk abstraction layer, which is a key technique for the efficient implementation of VCs.

In the rest of the paper, we introduce VirtuaLinux and Virtual Clusters (Sec. 2); we introduce the design and implementation of VirtuaLinux disk abstraction layer (Sec. 3). Then, we briefly describe VirtuaLinux VC facilities and tools (Sec. 4). Finally, we discuss VirtuaLinux Virtual Clusters performances (Sec. 5).

0-7695-3089-3/08 $25.00 © 2008 IEEE
DOI 10.1109/PDP.2008.86

IEEE computer society

2. VirtuaLinux

VirtuaLinux is an open-source Linux meta-distribution aiming at the following goals:

Disk-less Cluster Support Fragility due to disk-on-blades avoided by supporting a disk-less cluster architecture. Disks are replaced with a set of storage volumes, i.e. abstract disks implemented via an external SAN that is accessed via suitable protocols.

Master-less Cluster Support The single point of failure effect is avoided by removing the master from the cluster. Master node features, i.e. the set of services implemented by the master node, are categorised and made redundant by either active or passive replication in such a way they are, at each moment, cooperatively implemented by the running nodes.

Virtualized Cluster Support Both management flexibility and resilience to configuration errors are improved by means of transparent node virtualization. A physical cluster may support one or more VCs that can be independently managed without affecting the configuration of the underlying physical cluster. VCs can run a guest OS (either a flavour of Linux or Microsoft Windows) that may differ from the host OS, governing physical cluster activities.

These goals are achieved independently through solutions that have been designed to be coupled, thus to be selectively adopted. A suite of tools, included in VirtuaLinux, enable the boot, the installation, the configuration and the maintenance of a cluster exhibiting the previously described features. VirtuaLinux is currently targeted to AMD/Intel x86 64-based nodes, and includes one or more Linux distributions, currently Ubuntu Edgy 6.10 and CentOS 4.4/5.5; an install facility able to install and configure included distributions according to the above-mentioned goals; a recovery facility able to revamp misconfigured nodes; a toolkit to manage VCs and one or more pre-configured VC images, currently Ubuntu Edgy 6.10 and CentOS 4.4/5.5.

This paper focuses particularly on the virtualization support; we refer back to VirtuaLinux white paper [1] for a detailed description of other VirtuaLinux features, such as disk-less cluster boot and master-less services configuration.

2.1. Virtual Clusters

The virtualization of physical resources of a computing system to achieve improved degrees of sharing and utilisation is a well-established concept that goes back decades [9, 18, 23]. In contrast to a non-virtualized system, full virtualization of all system resources (including processors, memory and I/O devices) makes it possible to run multiple OSes on a single physical platform. A virtualized system includes a new layer of software, called a Virtual Machine Monitor (VMM). The principal role of the VMM is to arbitrate access to the underlying physical host platform resources so that these resources can be shared among multiple OSes that are guests of the VMM. The VMM presents to each guest OS a set of virtual platform interfaces that constitute a Virtual Machine (VM).

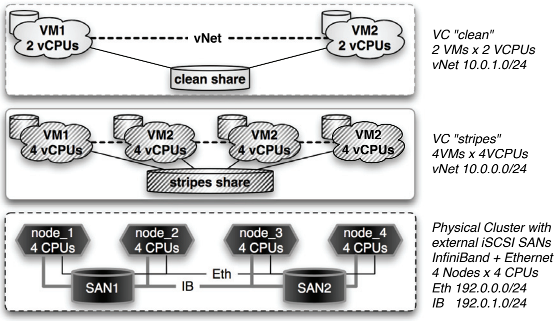

Figure 1. A physical cluster running two VCs.

By extension, a Virtual Cluster (VC) is a collection of VMs that are running onto one or more physical nodes of a cluster, and that are wired by a virtual private network. By uniformity with the physical layer, all VMs are homogeneous, i.e. each VM may access a private virtual disk and all VMs of a VC run the same OS and may access a shared disk space. Different VCs may coexist on the same physical cluster, but no direct relationship exists among them, apart from their concurrent access to the same resources (see Fig. 1). VCs bring considerable added value to the deployment of a production cluster because they ease a number of management problems, such as: *physical cluster insulation* and *cluster consolidation*. Physical cluster insulation ensures that crashes or system instability due to administration mistakes or cursoriness at the virtual layer are not propagated down to the physical layer and make no security or stability impact on the physical layer. Virtualization is used to deploy multiple VCs, each exploiting a collection of VMs running an OS and associated services and applications. Therefore, VMs belonging to different VCs may exploit different OSes and applications to meet different user needs. The main drawback of virtualization is overhead, which usually grows with the extent of hardware and software layers that should be virtualized.

Irrespectively of the particular VM technique adopted [23], VCs pose new challenges of disk management due to the need to support the creation and management of VCs on a physical cluster. In particular, *their creation and installation should be supported in time/space-efficient way*, since these operations are supposed to be frequently executed.

3. VirtuaLinux Storage Architecture

VirtuaLinux uses EVMS (Enterprise Volume Management System) to provide a single, unified system for handling storage management tasks, including the dynamic creation and destruction of volumes, which are an EVMS abstraction that are seen from the OS as disk devices [6, 19].

VirtuaLinux natively supports disk-less clusters, thus all the cluster permanent storage (at least OS-related files) are supposed to be stored in a iSCSI-attached[1] external SAN[2], usually composed by a fast and robust RAID[3]. The external SAN should hold a distinct copy of the OS for each node. At this end, VirtuaLinux prepares, during installation, one volume per node and a single volume for data shared among nodes. As we shall see later, several other volumes are used to realise virtual cluster abstraction. Volumes are formatted with an OS specific native file system (e.g. ext3) whereas shared volumes are formatted with a distributed file system that arbitrates concurrent reads and writes from cluster nodes, such as the Oracle Concurrent File System (OCFS2) or the Global File System (GFS).

Volumes are obtained by using the EVMS snapshot facility (see Sec. 3.1). A snapshot represents a frozen image of a volume of an original source. When a snapshot is created, it looks exactly like the original at that point in time. As changes are made to the original, the snapshot remains the same and looks exactly like the original at the time the snapshot was created. A file on a snapshot is a reference (at the level of disk block) to its original copy, and thus does no consume disk space while the original and its snapshot copy remain identical. A file is stored in the snapshot, and thus consumes disk space only when either the original or its snapshot copy is modified. Indeed, snapshot creation is quite a fast operation.

The snapshot technique is usually used to build on-line backups of a volume: the accesses to the volume are suspended just for the (short) time of snapshot creation; then the snapshot can be used as on-line backup, which can be kept on-line either indefinitely or just for the time needed to store it on a different storage medium (e.g. tape). Multiple snapshots of the same volume can be used to keep several versions of the volume over time. As we shall see in Sec. 3.2, the management of a large number of snapshots requires particular care in current Linux systems.

VirtuaLinux installs an original volume with the selected OS distribution (called the *default*), and then creates n identical snapshots. Each node of the cluster uses a different snapshot as the root file system. Once snapshots have been made accessible *(activated)*, the content of both original and snapshots can evolve along different paths; as they are independent volumes. However, in the case of cluster management, snapshots have several advantages as compared to independent volumes:

- *Fast creation time*. Assume an n-node cluster is to be installed. Since each node of the cluster requires a private disk, n independent volumes should be created at installation time starting from the same initial system distribution (e.g. CentOS system image). These volumes are physically stored in the same SAN due to the disk-less architecture. Creating these volumes by a standard copy loop may be extremely expensive in term of time since a complete Linux distribution should be installed n times.[4] Observe that a similar amount of time should be spent for the creation of each new VC. Snapshot usage drastically decreases volume creation time since volume content is not copied but just referenced at the disk block level. As an example, a snapshot of 10 GBytes volume can be created in a few seconds on a Giga-Ethernet attached SAN.

- *Reduced disk space usage*. In the general case, a snapshot requires at least the same amount of space as the original volume. This space is used to store original files in the case they are changed in the original volume after snapshot creation time, or new data stored in the snapshot that did not exist in the original volume at snapshot creation time. However, VirtuaLinux uses snapshots in an original way: the original volume holds the root file system of the Linux distribution, which does not change over time (when the original volume changes the snapshots is reset to reflect updates, see *centralised management*). Since data in the original volume is immutable to a large degree (OS files), a considerable amount of disk space is saved with respect to full data copy.

- *Device name independence*. EVMS ensures the binding of the raw device name (e.g. /dev/sda1) and logical volume name (e.g. /dev/evms/node1). Avoiding the use of raw device names when using iSCSI connected devices simplify the system configuration since they may exhibit different names on different nodes (this

[1] iSCSI is a network protocol standard, that allows the use of the SCSI protocol over TCP/IP networks. It enables many initiators (e.g. nodes) to access (read and write) a single target (e.g. SAN), but it does not ensure any coherency/consistency control in the case that many initiators access in read-write mode to the same partition [13].

[2] Storage Area Network.

[3] Redundant Array of Independent Disks.

[4] Estimated time depends on many factors, such as number of nodes, distribution size, DVD reader speed, SAN throughput. However, it can easily reach several hours even for small cluster configurations due the large number of small files that must be copied.

typically happens when a node has an additional device, e.g. an external DVD reader).

- *Centralised management.* A snapshot can be reset to a modified version of the original. Data that has been changed in the snapshot is lost. This facility enables the central management of copies, as for example for major system updates that involves all nodes. This facility is not strictly needed for cluster management since all snapshots can be changed, as they are different copies by using classical cluster techniques such as broadcast remote data distribution [25].

Since EVMS is a quite flexible and sophisticated management tool, the same goal can be achieved with different architectural designs, for example by using real volumes instead of snapshots with the EVMS cluster management facility. As discussed above, the VirtuaLinux design exhibits superior features with respect to alternative (and more classical) design options. The full description of EVMS functionality, which is outside the scope of this paper, can be found in [6, 19].

Note that VirtuaLinux uses the snapshot technique to provide a cluster with a number of independent volumes that can be efficiently created from a common template volume, whereas snapshots are usually used as transient, short-lived on-line backups. To the best of our knowledge, no other systems exhibit a similar usage of snapshots (and the consequent features). Indeed, in order to correctly exploit a different usage of snapshots, VirtuaLinux slightly extends EVMS snapshot semantics and implementation. This extension, which is described in the following sections, is correct with respect to EVMS snapshot semantics.

3.1. Understanding the Snapshot Technique

A number of different implementation approaches are currently adopted by vendors to create snapshots, each with its own benefits and drawbacks. The most common are *copy-on-write*, *redirect-on-write*, and *split mirror*. We briefly describe copy-on-write, which is adopted by EVMS; we refer back to the literature for an extensive description [10].

A snapshot of a storage volume is created using the pre-designated space for the snapshot. When the snapshot is first created, only the meta-data about where the original data is stored is copied. No physical copy of the data is made at the time the snapshot is created. Therefore, the creation of the snapshot is quite fast. The snapshot copy then tracks the changing blocks on the original volume as writes to the original volume are performed. The original data that is being written to is copied into the designated storage pool that is set aside for the snapshot before the original data is overwritten.

Before a write is allowed to a block, copy-on-write moves the original data block to the snapshot storage. This keeps the snapshot data consistent with the exact time the snapshot was taken. Read requests to the snapshot volume of the unchanged data blocks are redirected to the original volume, while read requests to data blocks that have been changed are directed to the "copied" blocks in the snapshot. The snapshot contains the meta-data that describes the data blocks that have changed since the snapshot was first created. Note that the original data blocks are copied only once into the snapshot storage when the first write request is received.

In addition to the basic functionality, EVMS snapshots can be managed as real volumes, i.e. data can be added or modified on the snapshot without affecting data on the original volume, provided that enough free space has been pre-allocated for the snapshot. Also, they can be activated and deactivated as standard volumes, i.e. mapped and unmapped onto UNIX device drivers. However, despite being standard volumes, snapshots have a subtle semantics with respect to activation due to copy-on-write behaviour. In fact, the system cannot write on an inactive snapshot since it is not mapped to any device, thus may lose the correct alignment with its original during the deactivation period. EVMS solves the problem by logically marking a snapshot for reset at deactivation time, and resetting it to the current original status at activation time.

3.2. Snapshots as Independent Volumes: an Original Usage

VirtuaLinux uses EVMS snapshots to provide a cluster with a number of independent volumes that can be efficiently created from a common template volume (original). Since snapshots cannot be deactivated without losing snapshot private data, they all should always be kept active in all nodes, even if each node will access only one of them.

Snapshots on Linux OS (either created by means of EVMS, LVM, or other software) are managed as UNIX devices through the *device mapper* kernel functionality. Although EVMS does not fix any limit on the number of snapshots that can be created or activated, current Linux kernels establish a hardwired limit on the number of snapshots that can be currently active on the same node. This limit comes from the number of pre-allocated memory buffers (in kernel space) that are required for snapshot management. Standard Linux kernels enable no more than a dozen active snapshots at the same time. This indirectly constrains the number of snapshots that can be activated at the same time, and thus the number of nodes that VirtuaLinux can support.

Raising this limit is possible, but requires a non-trivial intervention on the standard Linux kernel code. VirtuaLinux overcomes the limitation with a different approach, which does not require modifications to the kernel code. It leverages on the following facts:

- Since each snapshot is used as private disk, each snapshot is required to be accessible in the corresponding node only. In this way, each node can map onto a device just one snapshot.

- The status of an EVMS snapshot is kept on the permanent storage. This information is also maintained in memory in terms of available snapshot objects. This information is maintained in a lazy consistent way. Status information is read at EVMS initialisation time *(evms_activate)*, and committed out at any EVMS command (e.g. create, destroy, activate, deactivate a snapshot). While each snapshot can have just one status for all nodes on the permanent storage, it may have different status on the local memory of nodes (e.g. it can be mapped onto a device on a node, while not appearing on another).

- Snapshot deactivation consists in unmapping a snapshot device from the system, then logically marking it for reset on permanent storage. VirtuaLinux extends EVMS features with the option to disable EVMS snapshot reset-on-activate feature by way of a special flag in the standard EVMS configuration file. In the presence of this flag, the extended version of EVMS will proceed to unmap the snapshot without marking it for reset.

VirtuaLinux EVMS extension preserves snapshot correctness since the original volume is accessed in read-only mode by all nodes, and thus no snapshot can lose alignment with the original. One exception exists: major system upgrades, which are performed directly on the original copy of the file system, and that trigger the reset of all snapshots.

At the implementation level, the VirtuaLinux EVMS extension requires the patching of EVMS user-space source code (actually few lines of C code). Overall, VirtuaLinux extends EVMS semantics. The extension covers a case in which general conditions that have triggered the reset of a snapshot have been relaxed (avoids reset-on-activate) provided the original volume is not written. The extension ensures snapshot correctness. The described EVMS enables an original usage of the general snapshot technique.

4. Features of VirtuaLinux Virtual Clusters

VirtuaLinux implementation is arranged in a two-tier architecture: *VM implementation* layer and *VM aggregation* layer. The first one implements the single VM, currently the Xen VMM [2]. The second one aggregates many VMs in a VC, and dynamically creates and manages different VCs. This is realised via the *VirtuaLinux Virtual Cluster Manager* (VVCM). Overall, the VVCM enables the system administrator to dynamically create, destroy, suspend and resume from disk a number of VCs. The VCs are organised in a two-tier network: each node of a VC is connected to a private virtual network, and to the underlying physical network. The nodes of a VC are homogeneous in terms of virtualized resources (e.g. memory size, number of CPUs, private disk size, etc.) and OS. Different clusters may exploit different configurations of virtual resources and different OSes. Running VCs share the physical resources according to a creation time mapping onto the physical cluster. VCs may be reallocated by means of the run-time migration of the VM between physical nodes.

Each virtual node of a VC is implemented by a Xen VM that is configured at the VC creation time. Each virtual node includes: a virtual network interface with a private IP, a private virtual disk, a private virtual swap area and a VC-wide shared virtual storage. The virtualization of devices is realised by way of the standard Xen virtualization mechanisms.

4.1. Network Virtualization for VCs

Xen supports VM networking by way of *virtualized Ethernet interfaces*. These interfaces can be connected to underlying physical network devices either by way of *bridged* (OSI model layer 2) or *routed* (OSI model layer 3) networking. Bridging requires less setup complexity and connection tracking overhead as compared to the routing method. On the other hand, bridging impairs insulation among different networks on the same bridge, and it lacks flexibility since it can hardly be dynamically configured to reflect the dynamic creation and destruction of virtual networks. For this, VirtuaLinux currently adopts the routed networking.

VirtuaLinux sets up virtual networks in a simple manner: all nodes in the VC are assigned addresses from a private network chosen at creation time, and the VC does not share the same subnet as the physical cluster. In this way, the communications among physical and virtual clusters are handled by setting up appropriated routing policies on each physical node, which acts as a router for all the VMs running on it. Routing policies are dynamically set up at the deployment time of the VM. All VMs of all VCs can be reached from all physical nodes of the cluster and each VC can access to the underlying physical network without any master gateway node. Virtual nodes of a VC are simply VMs on the same virtual subnet. However, each virtual network is insulated from the others. The routing configuration is dynamic, and has a VC lifespan. The configuration is dynamically updated in the case virtual nodes are re-mapped onto the physical cluster.

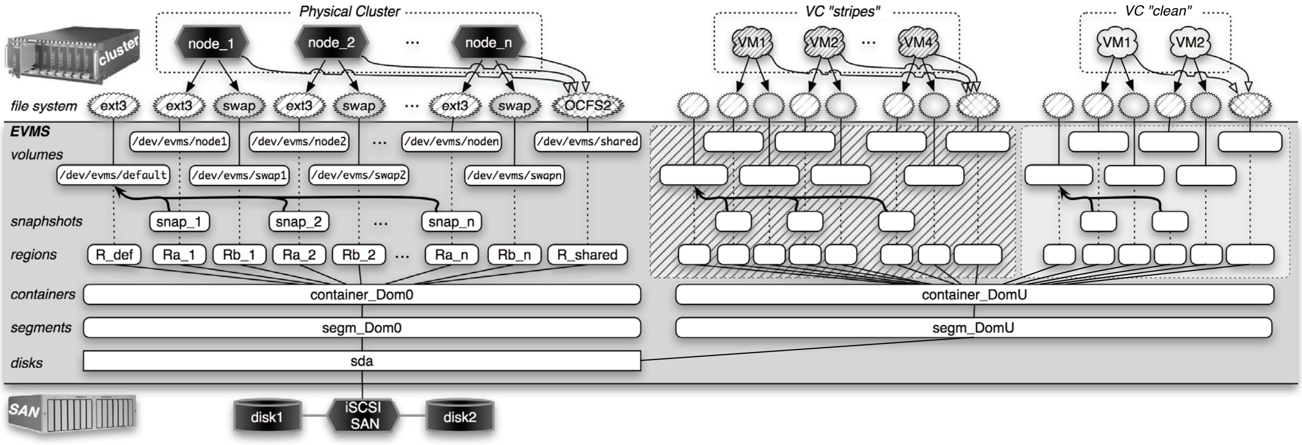

Figure 2. VirtuaLinux virtualized storage architecture.

4.2. Storage Virtualization for VCs

VirtuaLinux implements the storage abstraction layer described in Sec. 3, and exploits it for the storage of both the physical and the virtual nodes. The architectural view of VirtuaLinux is sketched in Fig. 2. In particular, the creation of a new VC proceeds in five stages; let us assume the new VC, called “stripes”, has 4 nodes:

- A standard EVMS volume `default_stripes` is created and installed with the chosen guest OS.
- 4 snapshots of `default_stripes` are created.
- 4 standard volumes are created and formatted as swap volumes.
- A configuration script is added to each snapshot, it will be called at the first boot of the VC to configure network addresses and other similar node-specific information.
- A configuration file holding “stripes” static information is created.

The process is repeated for the creation of each new VC. As shown in Fig. 2, the storage of each VC is derived from a common parametric template (that is also used for the physical storage). Once the VC is created, it can be started and managed via the VVCM, which completes the configuration file with dynamic information such as the virtual-to-physical mapping policy, and provides all the tools needed to deploy and run the VC. Observe that the virtual-to-physical mapping can be dynamically changed since all the created volumes are not bound to any particular physical node, but externally stored in a SAN.

As for the physical cluster, each VC comes with its own shared storage, which relies on OCFS2 [7] distributed file system to arbitrate concurrent read and write accesses from VC nodes. However, since Xen does not currently enable the sharing of many disks between VMs on the same physical node, the VC shared disk cannot be directly accessed from within virtual nodes. VirtuaLinux currently overcomes the problem by wrapping the shared storage with a NFS file system. At VC deployment time, each physical node involved in the deployment mounts the VC shared storage, which is in turn virtualized and make available to virtual nodes.

4.3. Management of the VCs

VirtuaLinux provides two strategies for virtual-to-physical mapping of VMs: *Block* and *Cyclic*. The first one aims to minimise the spread of VMs on the physical nodes. This is achieved by allocating on the physical node the maximum allowed number of VMs. The second one tries to spread the cluster’s VM across all the cluster’s physical nodes. The two strategies discussed can be coupled with two modifiers: *Strict* and *Lazy*. With the first modifier the deployment can be done only if there are enough free cores, with the second the constraint between the number of VM processors and physical cores is not taken into account at all. Notice that the mapping strategy of a VC can be changed after the first deployment provided it is the suspended state.

The VVCM (VirtuaLinux Virtual Cluster Manager) consist of a collection of Python scripts to create and manage the VCs. All the information about the virtual nodes such as the mapping between physical and virtual nodes and the state of each virtual machine are stored in a *database*. The information is maintained consistent between the launch of different clusters. A simple *command-line library* for the creation (*VC_Create* command), the activation (*VC_Control* command) and the destruction of the VCs (*VC_Destroy* command) is provided to the administrator. All the communications used for the staging and the execution of the VMs is implemented on top of the Secure Shell support (ssh). The *VC_Control* command relies on a simple *Virtual Cluster start-time support* to dynamically configure the network topology and the routing policies on the physical nodes for each VC.

5 Experimental Evaluation

Experimental data have been collected on a 4U-case Eurotech cluster hosting 4 high-density blades, each of them equipped with a two dual-core AMD Opteron@2.2GHz and 8 GBytes of memory. Each blade has 2 Giga-Ethernets and one 10 Gbits/s Infiniband NIC (Mellanox InfiniBand HCA). The blades are connected with an Infiniband switch. The external SAN is composed of a RAID0 (2 x 250GBytes SATA-RE Western-Digital WD2500YS) that is accessed via iSCSI (iscsitarget-0.4.14 [11]) on top of a Giga-Ethernet. The physical cluster has been installed two OSes that have been obtained by instantiating VirtuaLinux with Ubuntu and CentOS standard Linux distributions: i) a testbed installation Ubuntu Edgy 6.10 with Xen 3.0.1 VMM, Linux kernel 2.6.16 Dom0 *(Ub-Dom0)* and DomU *(Ub-DomU)*; ii) a reference installation CentOS 4.4, no VMM, Linux kernel 2.6.9 *(CentOS)*.

Three sets of micro-benchmarks have been used: the *LMbench* benchmark suite [14], which has been used to evaluate the OS performance; the *Intel MBI Benchmarks* [12] with *MVAPICH MPI* toolkit (mvapich2-0.9.8) [15], which has been used to evaluate networking performance; the Bonnie File System benchmark (bonnie1.03), which has been used to evaluate iSCSI and EVMS overhead.

According to LMbench, as expected, the virtualization of system calls has a non negligible cost: within both the privileged domain (Ub-Dom0) and user domain (Ub-DomU) a simple syscall pays a consistent overhead ($\sim +700\%$) with respect to the non-virtualized OS (CentOS) on the same hardware (while the difference between the privileged and the user domain is negligible). Other typical OS operations, such as fork+execve, exhibit a limited slowdown due to virtualization ($\sim +120\%$). However, as expected in a paravirtualized system, processor instructions exhibit almost no slowdown Overall, the OS virtualization overhead is likely to be amortised to a large extent in real business code.

The second class of experiments is related to networking. Figures 4 and 3 report an evaluation of the network latency and bandwidth, respectively. Experiments highlight that the only configuration able to exploit Infiniband potentiality is the one using user-space Infiniband verbs (that are native drivers). In this case, experiment figures are compliant with state-of-the-art performances reported in literature (and with CentOS installation, not reported here). Since native drivers bypass the VMM, virtualization introduces no overheads. As mentioned in Sec. 4.1, these drivers cannot be currently used within the VM (DomU), as they cannot be used to deploy standard Linux services, which are based on the TCP/IP protocol. At this aim, VirtuaLinux provides the TCP/IP stack on top of the Infiniband network (via the *IPoverIB*, or *IPoIB* kernel module). Experiments show that this additional layer is a major source of overhead (irrespectively of the virtualization layer): the TCP/IP stack on top of the 10 Gigabit Infiniband (*Dom0_IPoIB*) behaves as a 2 Gigabit network. The performance of a standard Gigabit network is given as reference testbed (*Dom0_GEth*). Network performance is further slowed down by user domain driver decoupling that require data copy between front-end and back-end network drivers. As result, as shown by *DomU_IPoIB* figures, VC virtual networks on top of a 10 Gigabit network, exhibits a Giga-Ethernet-like performances.

The third class of experiments is related to storage. According to Bonnie, the SAN sustains a raw 77 MBytes/s for block write and 77 MBytes/s for block read, respectively. The same SAN accessed via iSCSI+EVMS sustains 57 MBytes/s for block write and 87 MBytes/s for block read, respectively. The slowdown (-35% for write and -4% for read) is almost fully due to the remote access via iSCSI, since EVMS introduces no overhead for data read and fresh data write. On the contrary, due to the copy-on-write behaviour, a significant performance penalty is payed the first time that a file, existing in the original, should be overwritten ($\sim$7 MBytes/s). In this case, the linking between snapshot and original blocks on the disks should be broken, then the data can be written on the snapshot. Notice, however, that this penalty should be payed just the first time, as the rewrite operation exhibits the same performance in the two cases: after the first write the file on the snapshot became "fresh", i.e. fully independent from the original. Observe also, that in VC usage, snapshots differentiates at the first boot, thus the performance penalty is payed just once, and has no impact after the first boot.

Eventually, the proposed solution does not change the EVMS snapshot creation speed, that is easily more than two orders of magnitude faster than full data cloning (only meta-data is copied).

6 Related Works

Cluster Virtualization idea appears in several contexts as evolution of the single machine virtualization. In the cluster computing context, the bulk of the works focus either on the low-level recipe to build a platform-specific VC [4],

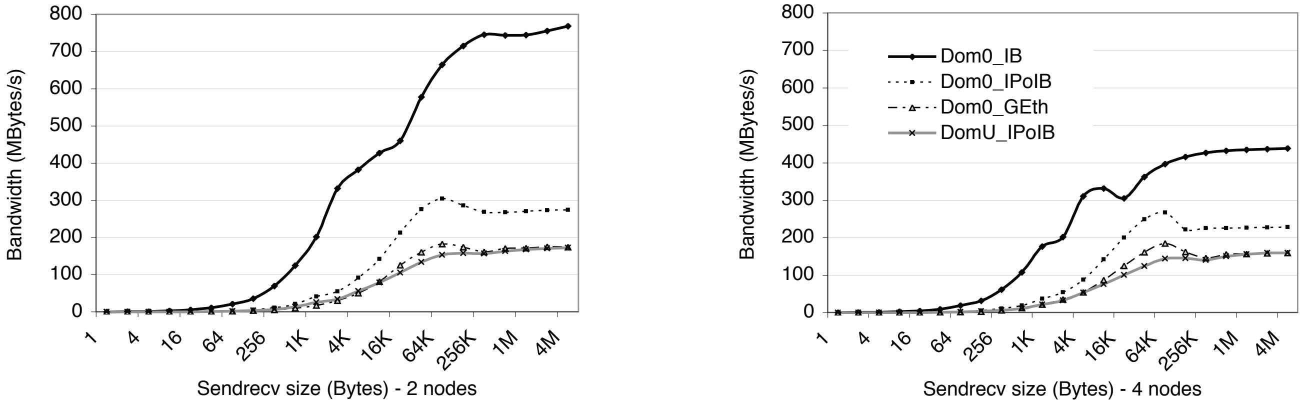

Figure 3. VirtuaLinux: evaluation of network bandwidth with the Intel MBI Benchmarks [12].

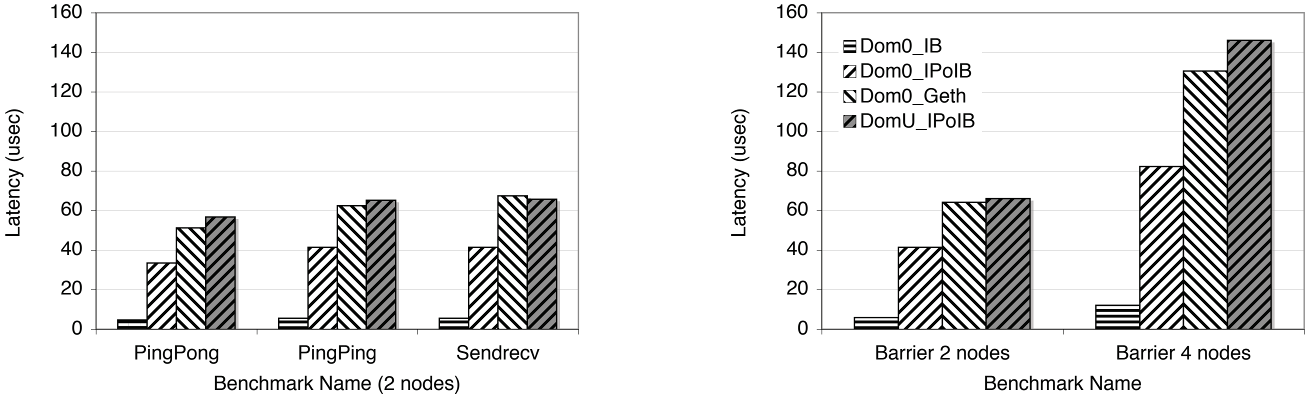

Figure 4. VirtuaLinux: evaluation of network latency with the Intel MBI Benchmarks [12].

or on the mechanisms needed to specify virtual nodes configurations, e.g. as XML files [3]. Many existing commercial and open solutions supports disk-less clusters [16, 5]. Many of them, use NFS to mount root directory while some of them natively supports disk-less clusters on top of iSCSI [24]. Almost all of them use explicit copies of node private data (either the full root directory or parts of it). These solutions greatly increase the complexity and the fragility of the installation since some of the standard OS packages should be reconfigured to write/read configuration and log files in different paths. VirtuaLinux differentiates from all of them since enables the transparent and efficient usage of standard Linux distributions, and natively includes the full support for Xen-based VCs. In the grid context, the idea evolved from the virtualization of the grid node aiming at addressing several issues, such as the support for legacy applications, the security against not trusted code and users, and the computation deployment independently of site administration. The support for VCs and their management tend to be integrated in the grid middleware, as an example in the Globus Toolkit 4 [8]. The Xenoserver project [22] is building a geographically distributed infrastructure as an extension of the Xen VMM.

VMware Lab Manager [26] appears affine to a VC manager. It enables to create a centralized pool of virtualized servers, storage and networking equipment shared across software development teams. Automatically and rapidly set up and tear down complex, multi-machine software configurations for use in development and test activities.

The possibility to use snapshots as virtual disks is mentioned in the LVM user guide [21], but only for a single platform. As discussed along the paper, even on a shared SAN, native LVM/EVMS snapshots can be hardly used "as is" due to scalability and security limitations. These limitations originate from the need to activate all snapshots on all physical nodes, independently on virtual-to-physical mapping and the status of the VC.

7 Conclusions

VirtuaLinux is a novel Linux meta-distribution aiming at installation and the management of robust disk-less high-density clusters. Among all features of VirtuaLinux, this paper has introduced virtual clustering architecture, features, and tools. These enable the dynamic and seamless creation and management of ready-to-use VCs on top of Xen VMM. Both the physical cluster and the VCs can be installed with a number of pre-defined OSes (e.g. Ubuntu Edgy 6.10 and CentOS 4.4) or easily extended to other Linux distributions. VCs managing tools can be easily extended to manage almost any guest Linux distribution by providing VC tools with a tarball of the OS and a simple configuration file.

VirtuaLinux introduces a novel disk abstraction layer, which is the cornerstone of several VirtuaLinux features, such as the time/space-efficient implementation of VCs. Experiments with VirtuaLinux has demonstrated the efficiency of the storage abstraction layer, and in general, the feasibility of the VC approach. Since VirtuaLinux design is largely independent from a particular VMM, its efficiency will naturally improve with virtualization technology evolution. In this regard, a test version of VirtuaLinux has been straightforwardly extended to support VCs by way of Linux kernel-based virtualization (KVM [20]), while the support of VCs based on proprietary guest OSes is currently ongoing (by way of WMware [26]).

VirtuaLinux is an open source software under GPL available at `http://virtualinux.sourceforge.net/`.

References

[1] M. Aldinucci, M. Torquati, M. Vanneschi, M. Cacitti, A. Gervaso, and P. Zuccato. VirtuaLinux design principles. Technical Report TR-07-13, Università di Pisa, Dipartimento di Informatica, Italy, June 2007.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proc. of the 9th ACM Symposium on Operating Systems Principles (SOSP'03)*, pages 164–177. ACM Press, 2003.

[3] R. Butler, Z. Lowry, and C. C. Pettey. Virtual clusters. In *Proc. of 18th Intl. Conf. on Systems Engineering (ICSEng)*, pages 70–75, Las Vegas, NV, USA, Aug. 2005. IEEE.

[4] A. de Vicente. *Building A Virtual Cluster with Xen*, July 2006. `http://www.clustermonkey.net/content/view/139/33/`.

[5] B. des Ligneris, M. Barrette, F. Giraldeau, and M. Dagenais. Thin-OSCAR : Design and future implementation. In *Proc. of the 17th Intl. Symposium on High Performance Computing Systems and Applications*, pages 261–265, Sherbrooke, Canada, May 2003.

[6] EVMS website. *Enterprise Volume Management System*, 2007. `http://evms.sourceforge.net/`.

[7] M. Fasheh. OCFS2: The Oracle Clustered File System, version 2. In *Ottawa Linux Symposium*, 2006.

[8] I. Foster, T. Freeman, K. Keahy, D. Scheftner, B. Sotomayer, and X. Zhang. Virtual clusters for grid communities. In *Proc. of the 6th Intl. Symp. on Cluster Computing and the Grid (CCGRID)*, pages 513–520, Singapore, May 2006. IEEE.

[9] R. P. Goldberg. Survey of virtual machine research. *Computer*, pages 34–45, June 1974.

[10] IBM. *Understanding and exploiting snapshot technology for data protection*, 2007.

[11] *The iSCSI Enterprise Target Project*, 2007. `http://iscsitarget.sourceforge.net/`.

[12] Intel Corporation. *Intel MPI Benchmarks: Users Guide and Methodology Description*, ver. 3.0 edition, 2007.

[13] iSCSI Specification. *RFC 3720: The Internet Small Computer Systems Interface (iSCSI)*, 2003.

[14] L. McVoy and C. Staelin. *LMbench: Tools for Performance Analysis*, ver. 3.0 edition, Apr. 2007.

[15] The Ohio State University. *MVAPICH: MPI over InfiniBand and iWARP*, 2007. `http://mvapich.cse.ohio-state.edu/overview/mvapich2/`.

[16] Penguin Computing. *Scyld ClusterWare HPC*, 2007.

[17] E. Pinheiro, W.-D. Weber, and L. A. Barroso. Failure trends in a large disk drive population. In *Proc. of the 5th USENIX Conf. on File and Storage Technologies (FAST'07)*, pages 17–28, San Jose, CA, USA, Feb. 2007.

[18] G. J. Popek and R. P. Goldberg. Formal requirements for virtualizable third generation architectures. *CACM*, 17(7):412–421, 1974.

[19] S. Pratt. EVMS: A common framework for volume management. In *Ottawa Linux Symposium*, 2002.

[20] Qumranet Inc. *KVM: Kernel-based Virtual Machine for Linux*, June 2007. `http://kvm.qumranet.com/kvmwiki`.

[21] Red Hat sourceware. *LVM2 Resource Page*, 2007. `http://sources.redhat.com/lvm2/`.

[22] D. Reed, I. Pratt, P. Menage, S. Early, and N. Stratford. Xenoservers: Accountable execution of untrusted programs. In *Proc. of the 7th Workshop on Hot Topics in Operating Systems*, Rio Rico, AZ, USA, 1999. IEEE.

[23] M. Rosenblum. The reincarnation of virtual machines. *Queue*, 2(5):34–40, 2005.

[24] *Stateless Linux*, 2007. `http://fedoraproject.org/wiki/StatelessLinux`.

[25] Sun Microsystems. *Sun Grid Engine*, 2007. `http://gridengine.sunsource.net/`.

[26] VMware Inc. *VMware website*, 2007. `http://www.vmware.com/`.