Virtual Distro Dispatcher: A Light-Weight Desktop-as-a-Service Solution

S. Cristofaro, F. Bertini, D. Lamanna, and R. Baldoni

Dipartimento di Informatica e Sistemistica "Antonio Ruberti"
"Sapienza" Università di Roma, Italy
{cristofaro,flavio.bertini,davide.lamanna,
roberto.baldoni}@dis.uniroma1.it
http://www.vdd-project.org

Abstract. Utility computing can occur at different levels. From Software-as-a-Service (SaaS) paradigm, Desktop-as-a-Service (DaaS) paradigm can be derived: desktops can be transformed into a cost-effective, scalable and comfortable subscription service. In VDD, desktop virtual machines are instantiated on a server and then provided to clients as a whole, on demand, across a network. Since the first release and publication, new features have been implemented and performance improvements achieved. As virtualization holds a critical role in the system, research and tests have been done for implementing the best virtualization solution. A comprehensive performance analysis is presented, depicting results that encourage to go on with the research and towards a real-life use. Operational costs analysis showed further economic and ecological advantages. The possibility to project operating systems not natively supporting the Xorg X11 has been introduced, opening the way to the projection of widespread though proprietary operating systems.

Keywords: XEN, UML, LTSP, Trashware, VDD-Project, Utility computing.

1 Introduction

Cloud computing architectures are rapidly spreading over the world of IT, supporting the idea of provisioning various computing capabilities"as-a-service", in a transparent way for users. Information is stored in servers on a network and cached temporarily on clients, such as desktops, entertainment centers, table computers, notebooks, wall computers, handhelds, etc. [7]. Reliable services are delivered to clients from next-generation data centers based on virtualization technologies. Some of the most relevant issues brought about by this paradigm are whether or not this is really feasible on a geographical scale, where network latency matters, and, more generally, whether or not a browser can really substitute every kind of computer application. Finally, big privacy issues rise: users data and work are given away in the hands of third parties, without any control and any real guarantee. Without necessarily dealing with these"cloudy" aspects,

D.R. Avresky et al. (Eds.): Cloudcomp 2009, LNICST 34, pp. 247–260, 2010.

it is always possible to reason about the more general concept of Utility computing, according to which computing resources, such as computation and storage, can be precisely metered and packaged, similarly to what happens with a traditional public utility, apart from the fact that the distribution of such a service happens to be in "The Cloud". Utility computing can occur at different levels. As long as applications are concerned, one talks about Software-as-a-Service (SaaS): applications are hosted as a service provided to users across a network (e.g., the Internet). If systems are concerned, one can talk about Desktop-as-a-Service (DaaS): desktops can be transformed into a cost-effective, scalable and comfortable subscription service. Desktops are instantiated on a server and then provided to clients on demand across a network. Virtual Distro Dispatcher [1] is a distributed system whose aim is to project virtual, fully operational operating system instances on terminals.

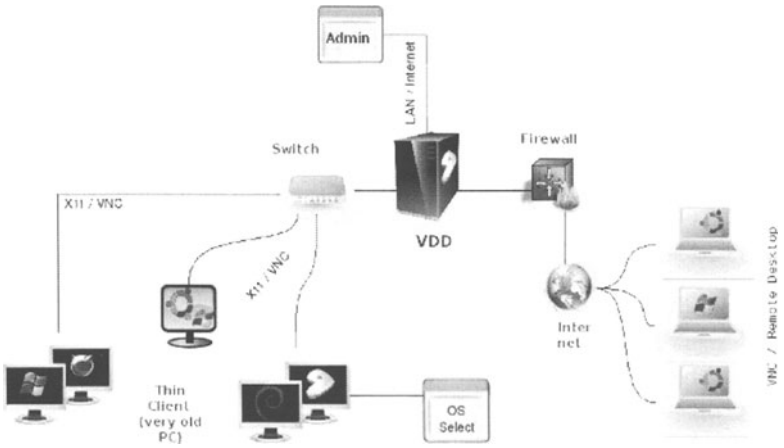

Fig. 1. Virtual Distro Dispatcher general scheme

The VDD architecture is represented in the Figure 1. More detailed information of the whole system are widely discussed in [1].

Client terminals can be obsolete PCs or energy saving thin clients (such as mini-ITX) managed by a powerful, multiprocessor (and possibly clustered) central system. The previous version of VDD, presented in [1], has got many limitations: Performances were still weak; Virtualization was performed only through UML[2] instances; Only Linux kernel based distributions could be projected on terminals. The new implementation of VDD hereby presented has focused in particular on performance improvements (as described in Section V). Moreover, operating systems other than Linux (e.g., Open Solaris, ReactOS, Microsoft Windows®...) can be accessed from terminals, thanks to the introduction of XEN[3] virtualization system. VDD gives users the possibility to enjoy their own favorite operating systems, including those that are not Open Source, possibly

at the same time, on each single thin client. It is important to remember (see [1] for details) that thin clients are interfaces to proper and isolated machines, that can be made to measure for whatever need and in whatever number (within server limits, of course). This is completely transparent to users, who, even from an obsolete machine, can select a particular machine with certain characteristics and then do absolutely everything they would do on such a machine as if it was physical and with its performance. Another dutiful remark regards licensing. Virtual Distro Dispatcher uses Open Source/GPL software and free communication protocols and it is released as Free Software. The infrastructure allows to run proprietary operating systems as guests and this is regulated by specific licenses, costs and limitations, that should be taken into account by VDD users.

2 Related Work

Using the taxonomy in [6], it is possible to identify three types of virtualized client computing (VCC):

1. **Application:** Encapsulating and isolating a specific application from its underlying host operating system and running it in the client environment, isolated from other applications running locally. Examples: Citrix Presentation Server (version 4.5+), Altiris Software Virtualization Suite, Thinstall, Microsoft SoftGrid, Trigence AE, Endeavors;
2. **Desktop:** Using virtualization to decouple the client environment (including operating system, application and data) from its host hardware and isolating it from other software or systems running on the client. It can be server-hosted or client-hosted. Server-hosted examples: VMware VDI, Virtual Iron VDI, Citrix XenDesktop, Qumranet Solid ICE. Client-hosted examples: VMware ACE/Player/Workstation/Fusion, SWsoft Parallels, Kidaro Managed Workspace, Sentillion;
3. **Virtual user session:** Creating multiple user sessions on the server, within a single operating system, that can be accessed concurrently. Examples: Citrix Presentation Server, Microsoft Terminal Services, Sun Secure Global Desktop Software.

Within these three types of VCC, two delivery models can be adopted (again in [6]):

- **Remote interaction:** I/O operations between a client device and a server through specific (and sometimes proprietary) protocols and software;
- **Remote streaming:** delivering executable blocks of data from a server to a client device, through specific (and sometimes proprietary) protocols and/or software.

VDD is in between type 2 and 3, as desktop virtualization software is used to host multiple unique and isolated client environments aboard a single server (or a group of servers in a cluster). Interaction with these remote virtual desktops is performed through virtual user (graphical) sessions. VDD exploits network

transparency of X-Window-System: the machine where an application program (the client application) runs can differ from the user's local machine (the display server). X-Window-System clients run on virtual servers and create multiple user sessions within multiple virtual environments. X-Window-System display servers run on thin clients (terminals). VNC protocols can be used for OSs which lack of X11 server (e.g. Windows and ReactOS), so both delivery models listed above are available.

The need for multiple development environments, especially in research laboratories, but also in teaching or developing circumstances, made the study of heterogeneous systems integration quite important. The availability of different operating systems at the same time, give users the possibility to develop software applications and to test them in several environments directly from their terminal, pursuing software portability. Other products supplying for this kind of service started to be developed. For example, an interesting DaaS system, Cendio Thin Linc[1], that is a LTSP based architecture like VDD, allows users to access remote desktops from everywhere[2]. Another example is NoMachine NX[3], which virtualizes desktops over the Internet too. VDD's main advantage is that only Free/OpenSource Software has been used, this being one requirement of our research. Another advantage is the extreme lightness, as highlighted in Section V. Development of virtualization systems plays a fundamental role in our research, mainly for performance issues. This is highlighted in section III, where more related work on this matter is cited.

3 Virtualization

Virtualization holds a critical role in VDD, as it enables the possibility to run multiple and diverse operating system instances to be projected to each thin client. The present piece of research focused on performance issues, hence several considerations and tests have been done in order to choose the best virtualization solution.

Unfortunately, the x86 architecture is more difficult to virtualize with respect to others, due to the presence of some particular instructions, such as the ones related to memory segmentation [5]. Even though, its large diffusion stimulated the development of many techniques to overcome such architecture limitations.

One of the most used virtualization techniques is the *binary rewriting* (also known as binary translation) which consists in scanning the code of the running guest with the aim of intercepting and modifying privileged instructions in order to fit in the virtualization system. Therefore, there is no need to modify the operating system source code, since all changes are made at run-time. On the

[1] http://www.cendio.com/products/thinlinc

[2] VDD is focused on projecting different operating system instances in the same LAN at the moment. Dispatching Linux on terminals over the Internet is technically possible, but not considered as something to deal with, at the moment (see also Section VII).

[3] http://www.nomachine.com/

other hand, there is a loss of performance, especially where the code contains several privileged instructions. The most popular virtualization systems using binary rewriting are VMware[4] and VirtualBox[5].

Another important technique is *paravirtualization*. It modifies privileged instructions, but at compile time instead of run-time. Even though modifying the guest operating systems source code implies an extra effort, one may notice a considerable performance increase, getting very close to an unvirtualized system (see Section V). Xen is one of the most powerful and famous virtualization system using mainly such a technique.

A more recent solution is the *Hardware Assisted Virtualization*. The last generation of AMD and Intel CPUs, have been developed with different virtualization extensions for x86 architecture[6]. The main purpose of these extensions, is to speed up the whole virtualization process and to make it easier for x86. Performance are in between the binary rewriting and paravirtualization techniques.

The choice of the virtualization system is fundamental to make VDD as performant as possible. Since the previous version of VDD uses User Mode Linux to dispatch Linux on terminals (for that reason, it was possible to emulate only Linux distributions), in order to make the right choice of a valid alternative and to add new functionalities, it has been useful to delineate a new list of constraints for our purposes (Table 1):

C1 - Open Source Software
C2 - Support for OS guest virtualization other than Linux (e.g. Microsoft Windows®)
C3 - Quick and easy to restore
C4 - Symmetric Multi Processing (SMP) OS guest support
C5 - User level kernel execution
C6 - Integrated VNC Server

Table 1. List of main VDD constraints

	C1	C2	C3	C4	C5	C6
VMWare		√	√	√[7]		√[8]
VirtualBox	√[9]	√	√			
UML	√		√		√	
Qemu	√	√	√	√		√
XEN	√	√[10]	√	√		√

[4] Since the version 5.5, Vmware Workstation also supports the Hardware Assisted Virtualization technique. The 6.0 version and above, supports also Linux guest paravirtualization.

[5] VirtualBox also supports Hardware Assisted Virtualization.

[6] AMD introduced the AMD-V extension (also known as Pacifica) whereas the technology used by Intel is called VT-x.

[7] VMware supports a maximum of two virtual processors per guest. VMware ESX Enterprise edition, supports up to four virtual processors.

Both Qemu and XEN satisfy most of the above main constraints, but tests showed that XEN is absolutely more performant than Qemu, mainly due to its use of paravirtualization for the guest OS supporting it[11].

4 Extension of Functionalities

The aim of VDD is to project virtual Operating Systems instances on thin clients. Unlike LTSP-only based architectures, offering only the host operating system to thin clients, VDD uses virtualized guest systems like sandboxes to prevent users from breaking into the central server. The isolation conditions produce an high security level both for the user and the overall system.

Since the beginning of the project, the utilization of UML allowed to run many different Linux distributions in user space. The next step was to introduce XEN as an alternative to UML. Although using XEN implies not to use completely user space virtualized systems, it is now possible to support much more operating systems other than Linux.

The introduction of advanced virtualization techniques made the system more performant as a consequence of both Hardware Assisted Virtualization and paravirtualization support. A further advantage comes from the possibility to assign many virtual CPUs to the guest systems, granting the symmetric multi processing to CPU-bound multi-threading tasks.

In the previous VDD version, UML was the only virtualization system, so the graphical subsystem was constituted only by Xorg X11 client/server model as the session projecting vehicle. The possibility to project operating systems not natively supporting the Xorg X11, brought to the need to set up a VNC client/server architecture. This has been possible thanks to the integration of a native VNC server inside XEN. In fact, a custom VNC client bash script has been added to LTSP [4] (running on Gentoo GNU/Linux) so that it could be possible to use it on thin clients, even if they are obsolete hardware.

Another strong point of this new release of VDD is to go over the technological gap due to the Trashware [8]. It is now possible to run a last generation operating system on an obsolete PC, like if it was running on a last generation computer, with negligible performance drop. For example, granting just for the sake of argument that it can be considered an actual bargain, it is now possible to run Microsoft Windows Vista®on a very old PC with a few memory resources.

5 Performance Analysis

A massive number of tests have been carried out in order to stress in depth system resources, such as CPU, memory, storage and network. For each such

[8] Only for the Server Edition.

[9] VirtualBox Open Source Edition has less functionalities respect of the closed source edition.

[10] XEN needs the VT-x or AMD-V technology to run unmodifiable Operating Systems.

[11] For non paravirtualizable OS guests, XEN uses a customized Qemu version.

system resource, one particularly significant test is hereby presented. The aim of the performance analysis is to understand as deeply as possible what happens at a system level in order to make then considerations about how this affects the desktop level. Tests have been performed on two architectures, 32 bit and 64 bit[12], using *LMbench* as the principal benchmark suite. In order to publish such tests, the LMbench license requires that the benchmark code must be compiled with standard gcc *compilation flags*. Furthermore, some standard applications, like Linux kernel compilation or John The Ripper benchmark have been used in tests. The testbed has got the following characteristics:

- Intel Core 2 Quad 6600
- RAM 4GB (667 Mhz Dual Channel)
- 2 SATA 500 GB Hard Disks (RAID 0)
- 2 1000Mbps Ethernet switches
- 10 diskless thin clients
- 14 1000Mbps Ethernet cards
- Cat. 6 FTP Ethernet cables

All tests have been carried out on the host system and inside the virtual machines, both for XEN and UML, in 32 and 64 bits both for the host and the guest systems. By *host*, the real host system is meant, i.e. an unpatched standard Gentoo Linux distribution, without any modification. Confusion should not be made with the XEN or the UML host, whose benchmarks are not relevant for comparisons. Hence, all tests have been performed on the standard host and within XEN and UML virtual machines. The following cflags have been used to compile[13] the analyzed systems:

Table 2. CFLAGS for VDD circumstances (host, UML and XEN)

Standard host system	-march=native -fomit-frame-pointer -pipe -O2
Host and guest UML systems	-march=native -fomit-frame-pointer -pipe -O2
Host and guest XEN systems	-march=native -fomit-frame-pointer -pipe -O2 -mno-tls-direct-seg-refs

Since the vanilla Linux kernel already includes virtualization patches[14] (paravirt-ops), tests have been performed both using the XEN source patches and vanilla Linux kernel, as regards guest tests.

In order to make a CPU-bound test, John The Ripper has been used. It includes benchmarks for all supported hashing algorithms. Such benchmarks are particularly suitable for the purpose of this investigation, since they make it possible to precisely evaluate the overhead introduced by virtual machines. Even though the two machines have a Core 2 Quad CPU, each test has been performed

[12] Two identical PCs have been used: one system has been compiled as 32 bit code, the other one as 64 bit code.

[13] Compiled using GNU gcc version 4.2.

[14] Since version 2.6.25 for 32 bit and since version 2.6.27 for 64 bit.

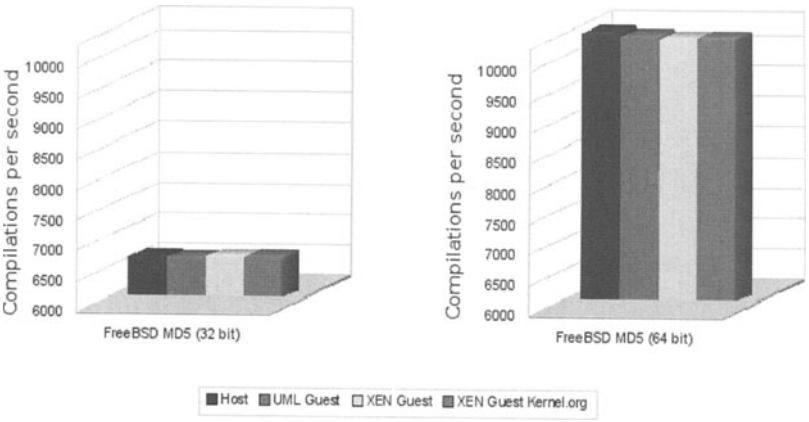

Fig. 2. Benchmark results for the John The Ripper test

only without Symmetric Multi Processing, in order to make comparisons with UML possible[15].

Thanks to paravirtualization, as expected, all results are quite close to each other. As it appears in the charts above (Figure 2), the overhead introduced by virtualization systems is quite unimportant. In any case, 64 bit systems proved to be far more performant.

As regards the LMbench memory mapping benchmark, an interesting difference between host and guest, especially for UML, can be noticed. The benchmark showed in the chart below is *bw_mmap_rd*, using the *open2close* option. This benchmark measures the bandwidth used to map a segment of a file to the memory and read it. Function *mmap()* belongs to a category of functions that is one of the hardest to be managed by virtual machines. This happens because virtual machines can not access physical memory directly. Hence, analyzing its behavior represents an excellent way to test system call management performed by paravirtualized systems and, in particular, to test how efficient is the hypervisor in managing it. As a matter of fact, this test is one of those in which Xen and, even more, UML loose more with respect to the host.

As a comment to the charts (Figure 3), all guest virtualized systems are sensitive to system call management. This is true especially for UML, due to the fact that it manages all system calls in user space, through a set of data structures, and this makes it quite slower than Xen. It is then possible to state that memory mapping management is the Achilles' heel of virtualized systems, even if Xen can cope with it better than others.

The next test is about filesystem latency. The test intends to verify the performance of virtualized systems in managing loop files (as in virtualized systems loop files act as virtual disks). In particular, the number of files created/deleted

[15] UML does not support SMP in skas4 mode. It was supported only it in TT mode, but TT mode is no longer supported.

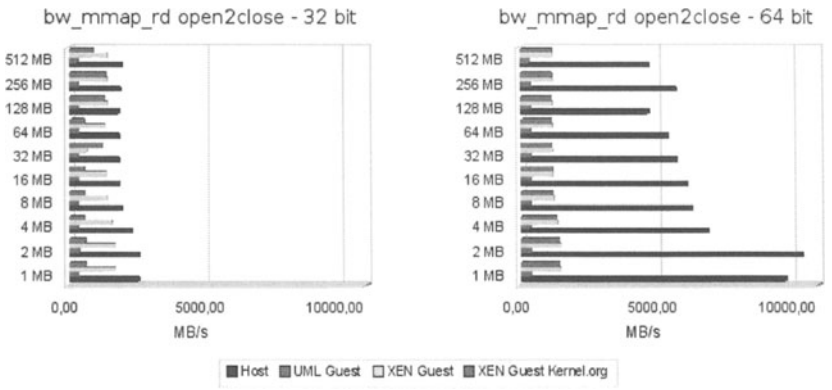

Fig. 3. Memory mapping benchmark results

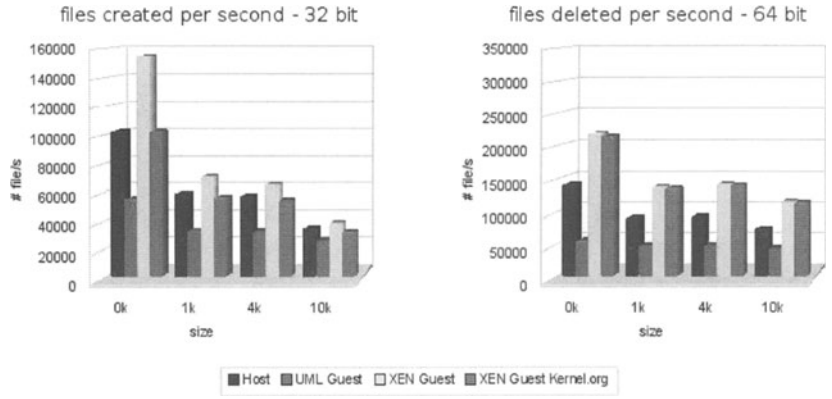

Fig. 4. Filesystem latency test results

per second is counted. The test has been repeated over files with different dimensions. Since guest systems are located into loop files, this may affect the test comparisons. In order to resolve this problem and to make tests comparable, a loop file have been generated also for the host system, which is so in exactly the same conditions of the guest. This test requires a destination directory where the system may create and delete files. So, each test has been performed inside each virtual machine. For the host system, the destination directory coincides with the loop file[16].

Results on Figure 4 show that the management of loop files in virtualized systems has reached an optimal level, especially for Xen. It is even better than the management of loop files made by the host system. This is because special functions have been developed in order to address such a critical issue. The test shown below is on memory again.

[16] All filesystems are ext3.

Fig. 5. Memory read/write speed test results

This test is useful to evaluate the overhead introduced for reading and writing in memory (Figure 5), after space is allocated, hence it does not take into account memory allocation, but only reading and writing speed. The test has been repeated with segments of memory with different size, in order to evaluate also the behavior of the system when cache is and is not functioning. Results show that the overhead is minimal and negligible, whatever the size is[17].

The next test is about performance decay due to virtual network cards with respect to physical network cards (Figure 6). The server is on a physical machine, while the client is on a virtualized machine. The two machines are connected via Gigabit Ethernet switches and cables. The test shows that virtual machines, on a physical network, do not introduce any significant overhead with respect to physical machines connected on the same network. In the picture below, the blue line represents the result of two physical hosts connected.

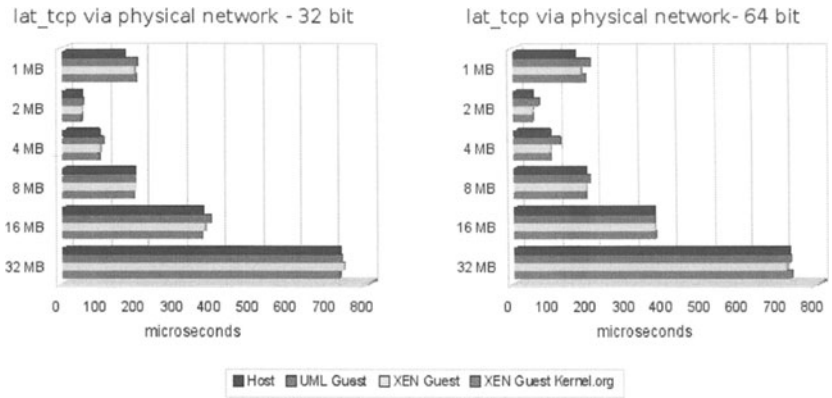

Fig. 6. Physical network latency test results

[17] Because of hierarchical memory (especially the 8MB L2 cache of the Q6600), results of reading small segments of memory are already in cache and hence obtained faster.

As it can be read in the man pages, lat_tcp is a client/server program that measures interprocess communication latencies. The benchmark passes a message back and forth between the two processes (this sort of benchmark is frequently referred to as a "hot potato" benchmark). No other work is done in the processes.

Another test could be the same of the previous, locally executed (i.e. both the client and the server are located within the localhost). There are no substantial differences for the systems involved in this test, apart from the fact that all data transfers are not conveyed through a physical local area network but through a virtualized network too. So, the whole network traffic is in the localhost.

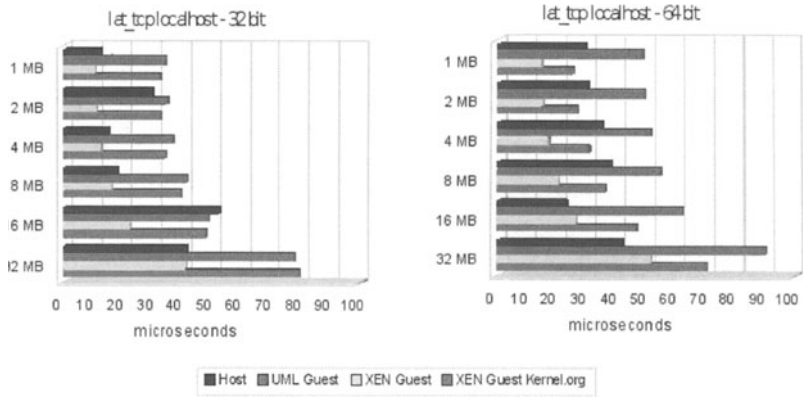

Fig. 7. Local host network latency test results

Test results on Figure 7 shows that the overhead is minimal and hence do not represent a bottleneck.

As a final remark, one can say that tests performed on VDD showed a negligible overhead introduced by the use of virtualization. This is true in particular for the tests hereby presented, which were selected based on the differences they are able to show in a more remarkable way with respect to others. The overhead may result significant only in particular situations (e.g., 3D graphic development), whereas performance at a desktop level is practically not affected. This is certainly encouraging for continuing the research, particularly if it succeed in showing more precisely the relation between system performance and desktop performance (see Section VII).

6 Operational Costs Analysis

VDD is an inexpensive solution born from the idea of Trashware [1],[8]. Research pushed forward from considering money saving for setting up a complete environment, to money saving for maintaining and operating it. For example, considering a LAN with 10 computers, three cases can be put to the test:

a) Buy 10 new PCs (no VDD)
b) Buy one new generation PC for the VDD server + 10 Trashware diskless thin clients for free
c) Buy one new generation PC for the VDD server + 10 mini ITX stations[18]

Solution *a* is far more expensive than the others, both for initial costs and for operational costs. In terms of initial costs, the cheapest solution is *b*, the only cost being the hardware for the VDD server management station, with money saving up to 83% with respect to *a*. This solution provides up to 18,5% for the energy saving[19]. Replacing obsolete and maybe cumbersome thin clients with mini-ITX terminals (solution *c*), money saving can be up to 72%. In this case, energy saving can arrive up to 71,4% (always with respect to *a*).

About the cost of a system upgrade, with solution *a*, a global operating system update has to be done on each PC, whereas with *b* and *c* solutions, an upgrade only involves the central server which is the VDD manager, since no operating systems resides in any diskless thin client. In this case, the whole system will result upgraded in one go.

A similar consideration can be done for hardware upgrade. Setting up VDD requires the central station to be powerful enough so that no significant overhead could influence thin clients utilization. As soon as the need for a hardware upgrade arises and/or more client stations are required, a more powerful central server could be needed. In regard to server-side hardware upgrade, it reflects to performance of all thin clients in one go, similarly to software upgrade. In regard to client-side hardware upgrade, instead, modifications for each thin client would be required. Economic-wise, this is not relevant, thanks to the reuse of hardware components refurbished through Trashware. This provides a practically unlimited amount of hardware dismissed too early by individuals or companies, and that are instead useful for building or upgrading VDD systems. In most cases, companies dismiss computers that are absolutely not obsolete as they consider[20]. Hardware reuse allows VDD thin clients to be upgraded and hence survive in pretty much all failure cases, by using the massive amount of spare hardware, produced by the current unsustainable production system, as a replacements resource.

7 Future Works

Setting up VDD may be rather demanding for people not so familiar with GNU/Linux and quite a high level of experience is required to manage all spare software components. One of the next step to further improve VDD is to develop a Graphical User Interface to make virtual machines dispatching as simple

[18] More generally, low energy systems such as mini/nano/pico-ITX.

[19] Considerations about energy cost analysis have been done consulting the http://www.eu-energystar.org/it/it_007c.shtml website. Each (thin client) station has been considered to be powered on for 6 hours per day.

[20] Social enterprises exist which work in refurbishing dismissed PCs thanks to Free Software. One of those is Binario Etico, www.binarioetico.org

as possible. Possible directions are: a web-based control panel, accessible from everywhere at any time and/or a host side interface to manage the whole environment from the central station. Code in Python has started to be written (having portability in mind).

As highlighted in Section V, it would be useful to explore more in depth relations between system level and desktop level regarding performance. Mapping application requirements to system specifications would help in designing and tuning the system for best desktop performance. Another interesting goal is to introduce High Availability Clusterization. VDD is managed by one central server at the moment. In [1], it was proposed to set up a HPC cluster like OpenMosix [8] to boost the core system. Unfortunately, HPC clustering does not have a wide interest any more, also due to the tremendous decrease of hardware price. Research interests are now focused on High Availability Clusters instead of HPC, in order to increase dependability and availability against failures [9].

As seen in Section 2, related work exists that consider dispatching desktops on the Internet an important characteristic. VDD can technically do that, even if this is not part of the research at the moment. It could be something to look at in the future, provided that the study on mapping system and network performance to desktop performance is carried out before. The high flexibility offered by the VNC protocol may allow to dispatch virtual Linux distributions over the Internet too. The only main difference is not to use obsolete computers as clients in this case, as data compression requires more performant PCs.

Privacy issues can easily be addressed by VDD, both at a local and at a global scale, simply by cyphering data. Although the whole system is quite safe, the utilization of encrypted volumes as filesystem partitions (using algorithms like AES-256), would give users the possibility to keep their data private and secure from intruders. Not even the administrator, who is in charge of managing such partitions, could be able to access data stored in them. This way, the well known privacy issue raised by cloud computing can be effectively addressed.

8 Conclusion

Intensely put to the test, VDD has proved to have wide margin to exploit as for system and network performance. VDD can open new frontiers of virtualization and distribution of resources by changing the way people resort to desktops. While the present paper was about to be finished, authors received news from NLnet foundation[21] regarding a request for funds to support the project, made by Binario Etico cooperative company[22]. NLnet decided to finance the project!

[21] http://www.nlnet.nl/ NLnet foundation financially supports organizations and people that contribute to an open information society, in particular it finances projects based on Free Software.

[22] http://www.binarioetico.org/ Binario Etico cooperative company sells products and services exclusively based on Free Software and reuse of obsolete PCs. It requested NLnet foundation for money to finance VDD project.

VDD emphasizes the importance of software over hardware. By using a new way of managing desktop environment software, VDD offers a technological point of view focused on ecology and saving, without renouncing to productivity and performance. Hardware development is closed to its saturation. VDD is the proof that software, in particular Free Software, can offer real ways to stimulate people creativity and reach new technological achievements.

References

1. Bertini, F., Lamanna, D., Baldoni, R.: Virtual Distro Dispatcher: A costless distributed virtual environment from Trashware. In: Stojmenovic, I., Thulasiram, R.K., Yang, L.T., Jia, W., Guo, M., de Mello, R.F. (eds.) ISPA 2007. LNCS, vol. 4742, pp. 223–234. Springer, Heidelberg (2007)
2. Dike, J.: User Mode Linux, April 22. Bruce Perens' Open Source Series, p. 352. Prentice Hall PTR, Englewood Cliffs (2006)
3. Chisnall, D.: The Definitive Guide to the XEN Hypervisor, 1st edn., p. 320. Prentice Hall PTR, Englewood Cliffs (November 19, 2007)
4. Linux Terminal Server Project, `http://www.ltsp.org`
5. Popek, G.J., Goldberg, R.P.: Formal Requirements for Virtualizable Third Generation Architectures. Communications of the ACM 17(7), 412–421
6. Rose, M.: (Industry Developments and Model) - Virtualized Client Computing: A Taxonomy (December 2007), `http://www.idc.com/getdoc.jsp?containerId=209671`
7. Hewitt, C.: ORGs for Scalable, Robust, Privacy-Friendly Client Cloud Computing. IEEE Internet Computing, 96–99 (September/October 2008)
8. Russo, R., Lamanna, D., Baldoni, R.: Distributed software platforms for rehabilitating obsolete hardware. In: OSS 2005: Proceedings of The First International Conference on Open Source Systems, pp. 220–224 (2005)
9. Cully, B., Lefebvre, G., Meyer, D., Feeley, M., Hutchinson, N.: Remus: High Availability via Asynchronous Virtual Machine Replication. In: Proceedings of the 5th USENIX Symposium on Networked System design and implementation, pp. 161–174 (Awarded Best Paper)