UMview: View-OS implemented as a System Call Virtual Machine [‡]

Renzo Davoli
University of Bologna
renzo@cs.unibo.it

Michael Goldweber
Xavier University
mikeyg@cs.xu.edu

Ludovico Gardenghi[§]
University of Bologna
gardengl@cs.unibo.it

1 Motivation

One component of the Virtual Square framework[12, 3, 4], is View-OS. Traditional OS's implement the *global view assumption*: any two processes running on a computer share the same view of their execution environment. This "global" view includes the meaning of pathnames, network stacks, routing rules, devices etc. While processes may have different permissions to access these resources, the naming scheme is the same. View-OS removes the global view assumption. Each process in View-OS has its own view of the execution environment, i.e. its own view of the networking system, file systems, existing devices, inter-process communication, etc. Furthermore, process view redefinition in View-OS can be selectively applied to specific portions of a processes' view: e.g. a process can change its view of the file system, or only on a subtree of the file system.

UMview[1] is a user-mode implementation of key View-OS concepts as a partial, modular, system call virtual machine (SCVM). A SCVM is a process-virtual machine[10, 11] where processes run natively on the processor and just the system calls are virtualized. User-Mode Linux (UML)[5, 6], whose virtual monitor is an entire linux kernel loaded as a process, is the canonical example of a SCVM.

UMview is a *partial* SCVM since it is possible to provide a process with a view which is a mix of virtual and native entities. For example, UMview supports the mounting of a virtual filesystem. Afterwards, the processes' view of the filesystem is composed of a virtual subtree rooted at the mountpoint, in addition to the pre-existing filesystem. Unlike traditional kernel mounts, UMview mount effects are limited to the processes running in the virtual machine. Furthermore, view modification actions, like the mount operation example, can be nested.

UMview is a *modular* SCVM. Loadable modules provide suitable abstractions for supporting specific virtualizations: e.g. file systems, devices, networking systems, etc.

UMview provides a flexible and general purpose support for virtualization. Using the View-OS concepts implemented as UMview modules one can "unify" several pre-existing models of virtualization.

- chroot system call;
- system call interposition [14, 7, 9];
- fakeroot utility;
- virtual file systems [8, 15];

2 Current State of the Project

UMView runs in user-mode on vanilla GNU/Linux 2.6 kernels and supports binary compatibility with standard Linux executables. Though not a complete View-OS implementation, UMview is a working and usable proof-of-concept. Its current features provide highly useful extensions to the traditional semantics of the underlying system.

Each specific virtualization is managed by a module, which can be loaded/unloaded at run time. Currently, there are six modules:

umfuse allows mounting of FUSE file systems (including source code compatibility with existing modules) without the need for kernel FUSE support. Disk images, network mounts and other file systems can be accessed by the user with no need for specific tools. **umfuse** also supports nesting, i.e. one can mount a disk image which resides on a second umfuse-mounted disk image or file system.

umlwipv6 extends the LWIPv6 user-level TCP/IP stack allowing for network virtualization on a per-process basis; a process can see its own network stack (with separate IP address and routing rules). Connections with the real world are usually made via a VDE [2] connection. Module composition is also supported; a virtual network can be used to mount remote file systems with **umfuse**.

umdev implements virtual devices and adds support for I/O control system call virtualization. Each special file or group is managed by a sub-module. One of them is **umdevmbr** for hard disk MBR virtualizations. Users can mount a disk image, partition it (accessing to virtualized copies of block devices like `/dev/hda`), create file systems on the new partitions (`/dev/hda1, ...`) and mount the resulting file system with **umfuse**.

viewfs performs various kinds of transformations on the file system namespace as seen by the process. Using **viewfs** one can *hide* portions of the file system; e.g. to isolate the effect of malicious/ill-behaving software on key data). It can also implement the *copy-on-write* abstraction, allowing processes to operate on read-only portions of the real file system in a safe manner.

umbinfmt is a user-mode implementation for binfmt support which lets users choose a different "interpreter" for an executable depending on the file extension or specific file pattern. For instance, QEMU [1] can be used to run a Linux/i386 executable from a Linux/ppc command shell.

Finally UMview also includes some optional kernel patches which aid UMView performance. Since these optimizations can also be exploited by other virtual machines (e.g. UML) these patches are currently under consideration to be included in the official Linux kernel.

References

[1] F. Ballard. Qemu project home page. http://fabrice.bellard.free.fr/qemu/.

[2] R. Davoli. Vde: virtual distributed ethernet. In *Proceedings of Tridentcom 2005*, Trento, 2005.

[3] R. Davoli. Virtual square. In *Proceedings of OSS2005. Open Source Software 2005*, Genova, 2005.

[4] R. Davoli and M. Goldweber. Virtual square in computer science education. In *Proceedings of ITiCSE05. Conference on Innovation and Technology in Computer Science Education*, Lisbon, 2005.

[5] J. D. Dike. User-mode linux. In *Proc. of 2001 Ottawa Linux Symposium (OLS)*, Ottawa, 2001.

[6] J. D. Dike. Making linux safe for virtual machines. In *Proc. of 2002 Ottawa Linux Symposium (OLS)*, Ottawa, 2002.

[7] T. Garfinkel, B. Pfaff, and M. Rosenblum. Ostia: A delegating architecture for secure system call interposition. In *Proc. Network and Distributed Systems Security Symposium*, February 2004.

[8] P. Miller. The plastic file system 1.9. http://plasticfs.sourceforge.net/.

[9] N. Provos. Improving host security with system call policies. In *12th USENIX Security Symposium*, Washington, DC, August 2003.

[10] J. Smith and R. Nair. *Virtual Machines: Versatile Platforms for Systems and Processes*. Morgan Kaufmann, 2005.

[11] J. E. Smith and R. Nair. The architecture of virtual machines. *IEEE COmputer*, 38(5):32–38, May 2005.

[12] The V^2 project team. Virtual square home page. http://www.virtualsquare.org.

[13] The View-OS project team. View-os savannah project. http://savannah.nongnu.org/projects/view-os.

[14] D. A. Wagner. Janus: an approach for confinement of untrusted applications. Technical Report CSD-99-1056, University of California, Berkeley, 12, 1999.

[15] C. P. Wright and E. Zadok. Unionfs: Bringing File Systems Together. *Linux Journal*, pages 24–29, December 2004.

[‡]Submitted for OSDI 2006 poster session. The poster presentation includes a live demo of the software, some kind of network access is preferred during the presentation. This work was partially supported by the WebMinds FIRB project of the Italian Ministry of University, Research and Education.

[§]student member of the research team

[1]Released under GPLv2 is available at the Savannah repository of the View-OS project [13].