Namespace Management in Virtual Desktops

DUTCH T. MEYER, JAKE WIRES, NORMAN C. HUTCHINSON, AND ANDREW WARFIELD

Dutch Meyer is a PhD student, under the supervision of Andrew Warfield, at the University of British Columbia. His research investigates the impacts of virtualization on network-available storage systems.
dmeyer@cs.ubc.ca

Jake Wires received his MS in computer science from the University of British Columbia. He currently works in the Datacenter and Cloud Division at Citrix, where his focus is storage virtualization.
Jake.Wires@Citrix.com

Norman Hutchinson is an associate professor at the University of British Columbia. His research interests center on programming languages and operating systems, with particular interests in object-oriented systems, distributed systems, and file systems.
norm@cs.ubc.ca

Andrew Warfield is an assistant professor at the University of British Columbia. He advises students on a wide range of topics, including virtualization, storage, and security.
andy@cs.ubc.ca

Even as virtualization has promised to ease cluster scale and management, it presents system administrators and storage system designers with opaque blobs of data that represent entire virtual volumes. In these environments, application and file-level semantics are abandoned long before data reaches the disk. Our research borrows from past work and is creating virtual storage interfaces that preserve file-level information in order to improve the management and efficiency of storage.

Virtualization has been widely used to reduce operational costs of mid- and large-scale server farms. Now it is making headway in desktop computing, where it has played a central role in recent efforts to migrate users from individual workstations to centrally administered servers. Virtual Desktop Infrastructure (VDI) is the latest manifestation of the well-known thin-client paradigm. It attempts to lure end users—who have previously been reluctant to embrace thin clients—by providing a computing environment almost identical to the familiar desktop PC. There are good reasons to believe this approach is working.

Gartner predicts that 40 percent of all worldwide desktops—49 million in total—will be virtualized by 2013 [2]. Already, many organizations have deployed VDI to tens of thousands of users [6]. This surge is being driven by administrators who have long seen the value of centralizing PC resources. They find that a significant economy of scale comes from the reduced operating costs provided by a VDI environment. There are, however, big challenges posed by such large and centralized installations, particularly with respect to storage.

In current VDI implementations, virtual disks are stored as opaque files on a central network server. File formats like Microsoft's VHD and VMware's VMDK encapsulate entire disk images using a read-only base image (or Gold Master) as a template disk. Modifications to the base image are stored in one or more separate overlay images allocated for each VM. This allows the rapid creation of new VMs with minimal initial overhead. But as VMs mature, their overlay images diverge, leading to increased storage consumption and maintenance burden. The block level approach used by these file formats, while simple to implement, lacks the contextual information necessary to begin addressing this divergence problem. Administrators are presently faced with the choice of either allowing images to diverge without bound, resulting in a serious management problem for tasks like upgrade, or resetting images to the original master at daily or weekly frequencies, which frustrates users who desire to install their own software.

This information-poor block interface also extends through much of the storage stack in enterprise VM environments. In a traditional PC the transformation from file to block requests occurs very low in the stack, so many storage features operate at the file level. However, in virtualization environments this transformation occurs at the top. Below the guest VM's block level, the VMM will map the virtual drive to a file format for virtual disks. Since a shared storage system is required for live VM migration and recovery in case of a physical server failure, the block requests must then travel over the network. They are processed by a centralized storage system that aggregates many physical drives, often storing the images on yet another file system. In total, an enterprise virtualization storage stack will have easily twice the distinct layers of a desktop PC, most of which are unaware of the original file semantics.

Semantics Lost

This loss of semantics limits file-oriented performance optimizations. For example, it is often the case that different VMs on the same physical host read and store identical files that happen to be at different logical disk offsets. Common storage optimizations around caching, deduplication, and placement—often implemented within OS and file system code with the benefit of object boundaries—must be approximated at the block layer.

Block semantics also diminish the administrator's ability to administer. In time-sharing systems, administrators could see user files and their accesses. If a configuration file was incorrect, it could be inspected and even changed. If a file management policy was not being followed, it could be detected directly. Administrators could scan the whole system for all files of a given type or name. The corresponding view in a contemporary virtualization system is a stream of block requests passing to an opaque virtual disk file.

This can make simple tasks, such as changing a user's security settings in Internet Explorer, unnecessarily complicated. The administrator can use Remote Desktop or Terminal Services to modify the machines directly, but must access each VM individually. With scripts they may do the job faster, but this requires knowledge of specialized syntax. If the VMs can be turned off, the administrator can mount the disks for inspection. Or perhaps she could email the VM's owner and ask politely for help. These restrictions are largely consequences of using opaque containers and protocols. They seem ridiculous, given that the files are already being hosted on a single shared storage system.

Users do not directly see these layer intricacies, of course, but they also don't get many explicit benefits. Instead, they are forced into an anachronism—PC-era isolation, despite mainframe-style consolidation. Consider file sharing in this environment: users can copy files to a network drive, create a file server on their local VM, or email files as attachments. Those options seem natural for a PC, but in a VDI any shared data is already hosted by a dedicated file server. The barriers to collaborating on this file are vestiges of an era when hard drives were physically isolated—and virtual disks preserve these barriers without offering any real benefits. A better approach would be to share the file without creating copies and without the complexity and overhead of creating what is in effect a proxy file server.

Obviously, today's file systems were not designed for use in virtual environments. But what would a new, virtualization-aware file system look like? We feel that some very good ideas can be reappropriated, refurbished, and redeployed from past research to help address these issues.

Namespace Composition

Many of the these problems stem from the forfeiture of file semantics at the top of the predominantly block-oriented virtual storage stack. However, there's no fundamental reason that most of the storage stack can't instead use a file interface. File-based network protocols like CIFS and NFS are in widespread use, and virtual disk management based on a file interface was introduced with Ventana in 2006 [3].

Like Venti [4], which inspired it, Ventana used a single global store for all files. Individual disk images were created by selecting the necessary files from this pool, and shared access was protected with copy-on-write. Conceptually, this composition could be considered similar to a very fine-grained use of UnionFS. Unfortunately, the Ventana implementation never saw much use or distribution.

Systems like Ventana require that the file interface extend all the way from the guest operating system to the network storage system. In practice, there are technical limitations that make this difficult. Most notably, the Windows boot process requires a block device, which precludes using a file interface. However, this problem is not impossible to overcome. Linux already provides NFS boot, which would be a sufficient solution for Windows. In our own lab we use a custom Windows file system driver to transition to a CIFS interface during boot, which has much the same result. Although this approach requires synthesizing a block interface for the early boot stages, it is much simpler than recreating file semantics from a stream of block requests [1].

Whether one uses CIFS, NFS, or another protocol in place of a block interface, the benefits are significant. At the VM level, it removes the need for any type of block-level processing. The VMM benefits from the file interface, because it opens up opportunities for caching when multiple VMs are reading from the same files. The network interface to centralized storage can also be file-based, possibly NFS or CIFS, which is easier to reason about than iSCSI and available on more affordable hardware. Finally, the cluster administrator is put back in the position of dealing with file access and management. This helps in technical administration, such as troubleshooting a client misconfiguration and managing diverging disks. It opens the door to replacing inefficient per-client services, such as virus scanners, with centralized alternatives. This could be used to solve the "Antivirus Storm" problem, where a number of idle clients, unaware that they are all sharing storage resources, engage their antivirus software and place stress on centralized storage.

A file interface would also help administrators simply understand what their system is doing. Often, it is too easy for an administrator to be unaware that a considerable portion of their storage resources is handling completely unnecessary tasks such as defragmenting virtual disks or writing IE temp files over the network to highly redundant and expensive storage. Given the current structure of block requests and opaque disk images, such waste can go unnoticed.

Virtual Directories

Restoring file semantics to the storage stack may not be enough. Many believe that today's file systems are already too complex to manage [5], so navigating a large cluster of such hierarchies would probably challenge an administrator. In other environments, virtual directories have been the subject of some interest in combating complex file hierarchies. For virtualization, we think the idea could be extended to provide even more benefits.

The virtual directory mechanism traces back to Gifford's 1991 paper proposing semantic file systems, and perhaps even earlier, to UNIX systems which first displayed devices through a file interface. In the current context, complex searches for files can be represented persistently as a virtual directory. This allows users to create directories that display semantic information rather than filesystem location information. As an example, users might want their music collection displayed in the file system as a directory containing all files from a certain artist, regardless of the hierarchical location of those files.

Combined with an enterprise-wide storage system like the one proposed above, this mechanism would be a powerful management and collaboration aid. For end users, this would provide support for three fundamental workflows.

First, for users operating on multiple VMs, the process of circumventing the unnecessarily strong barriers between VM file systems could be eliminated. Rather than creating a new file server or copying the file over a network, a user could merely request that the file be mapped into both file systems. Of course, there are complex notions of user-identity and access control that need to be addressed. Similar problems have been addressed in the past [7], although in different contexts, making this a ripe area for further research.

Second, to facilitate information finding, one could use virtual directories for persistent queries, such as "find me all spreadsheet files from the accounting group." Currently, one could search for such files, but the illusion of decentralized storage requires that users first locate the appropriate network servers and then aggregate results from multiple sources. Furthermore, persistent queries are more powerful than searches, because they can stay current with publish/subscribe notifications.

Third, virtual directories and namespace composition could work together to empower file publishers, while simplifying the steps required to collaborate on a file. Rather than sending an email to relevant parties containing a network address or copy of the file, a publisher could (with the help of a Microsoft Outlook plug-in) include a capability to access the file in an email. Shared access to this file could be coordinated with copy-on-write, writer locks, or integrated version control software. No doubt, each of these options should probably be available, since different collaboration models are appropriate for different files. In any case, this approach shifts the age-old problem of maintaining and merging multiple file copies from an ad hoc management approach to one that is consistent and centralized in a single enterprise-wide file system. Certainly, merging will occasionally be required, but that's okay. Most non-expert computer users are already familiar with the need to merge files, since they do this over email already. What they aren't aware of is the fact that other management options exist for this problem.

Virtualization administrators, similarly, would benefit greatly from virtual directories. Aggregating files from VM file systems into a single namespace could provide opportunities to view a user's files in terms of their similarity or dissimilarity to those of their peers. The latter may provide opportunities to locate misconfigured machines via outlier detection. The former would allow administrators to considerably collapse the large space of files in a large enterprise. Virtual files may also be useful: consider, for example, creating a master log for a cluster by reading Windows Events through the logging facilities of each VM and merging them. Again, these mechanisms provide new opportunities to diagnose problems or to catch warnings before they become problems.

Towards a Virtualization-Friendly File System

Cluster-wide virtualization is disruptive to internal network, compute, and storage infrastructure. However, corresponding changes have yet to propagate to our file systems. Our research experiences suggest to us that deep stacks without semantic information lead to misconfiguration and inefficiency. Namespace composition offers one organizing principle, but many issues remain. Simplicity and platform-agnosticity at the block level have served us well, but ensuring those traits in file- and object-based protocols is more difficult. There are also questions about layering in the storage stack. It's not yet clear how much functionality should be placed in the client file system. Alternatively, the VMM's role in hosting many guest file systems suggests performance benefits to co-locating similar VMs and providing features at that level. Then again, centralizing storage in back-end filers is appealing for simplicity.

For end users, there is already widespread awareness that we need better tools to organize and navigate data, but virtualization may be important in shaping those solutions. Virtualized desktops and datacenters act much like PCs, but their architecture is closer to time-sharing systems. We need to find a balance between isolation and ease of sharing in these environments. Even for individual users, creating VMs in order to isolate known-good OS and application configurations is beneficial. However, sharing and synchronizing files between these isolated systems is not easy or robust. With support for file sharing between VMs, virtual directories offer a compelling alternative. Semantic file organization may also improve our ability to find what we want among larger collections of file systems. However, that approach implies that namespace location is no longer a useful guide for the physical disk locations for our files.

While many open questions remain, we see great promise in these semantic-rich storage stacks and file system structures. Administrators need a file-oriented view of storage to efficiently understand and assist their users, while users may see benefits to novel file organizations and simpler work flows through explicit sharing. Storage is, for many good reasons, slow to change, but if we are to address the shifts in PC and cluster design, changes are on the way.

References

[1] A.C. Arpaci-Dusseau and R.H. Arpaci-Dusseau, "Information and Control in Gray-Box Systems," in *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles,* 2001, pp. 43–56.

[2] C. Pettey and H. Stevens, "Gartner Says Worldwide Hosted Virtual Desktop Market to Surpass $65 billion in 2013," March 2009: http://www.gartner.com/it/page.jsp?id=920814.

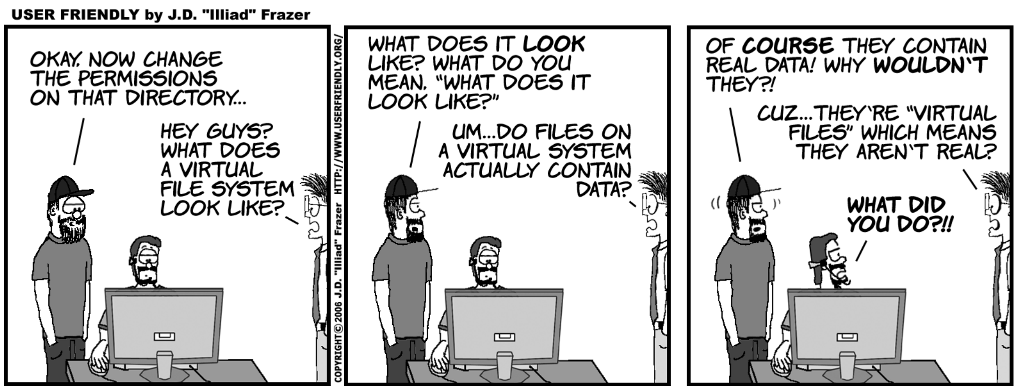
[3] B. Pfaff, T. Garfinkel, and M. Rosenblum, "Virtualization Aware File Systems: Getting Beyond the Limitations of Virtual Disks," in *3rd Symposium on Networked Systems Design and Implementation (NSDI '06),* 2006.

[4] S. Quinlan and S. Dorward, "Venti: A New Approach to Archival Storage," in *Proceedings of the Conference on File and Storage Technologies,* 2002, pp. 89–101.

[5] M.I. Seltzer and N. Murphy, "Hierarchical File Systems Are Dead," *12th Workshop on Hot Topics in Operating Systems (HotOS XII)*: http://www.usenix.org/event/hotos09/tech/full_papers/seltzer/seltzer.pdf.

[6] VMWare, Customer Case Studies by Product, November 2010: http://www.vmware.com/products/view/casestudies.html.

[7] E. Wobber, M. Abadi, M. Burrows, and B. Lampson, "Authentication in the Taos Operating System," in *Proceedings of the Fourteenth ACM Symposium on Operating Systems Principles,* 1993, pp. 256–269.


USER FRIENDLY by J.D. "Illiad" Frazer
OKAY. NOW CHANGE THE PERMISSIONS ON THAT DIRECTORY...
HEY GUYS? WHAT DOES A VIRTUAL FILE SYSTEM LOOK LIKE?
COPYRIGHT © 2006 J.D. "Illiad" Frazer HTTP://WWW.USERFRIENDLY.ORG/
WHAT DOES IT LOOK LIKE? WHAT DO YOU MEAN, "WHAT DOES IT LOOK LIKE?"
UM...DO FILES ON A VIRTUAL SYSTEM ACTUALLY CONTAIN DATA?
OF COURSE THEY CONTAIN REAL DATA! WHY WOULDN'T THEY?!
CUZ...THEY'RE "VIRTUAL FILES" WHICH MEANS THEY AREN'T REAL?
WHAT DID YOU DO?!!