VMTorrent: Virtual Appliances On-Demand

Joshua Reich, Oren Laadan, Eli Brosh, Alex Sherman, Vishal Misra, Jason Nieh, Dan Rubenstein
Department of Computer Science, Columbia University
reich,orenl,elibrosh,asherman,misra,nieh,danr@cs.columbia.edu

Categories and Subject Descriptors

D.4 [**Operating Systems**]: Miscellaneous

General Terms

Algorithms, Design, Performance

Keywords

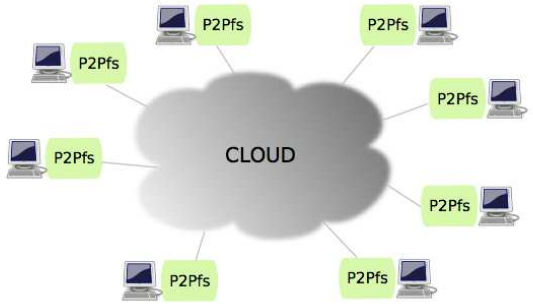
BitTorrent, Cloud Computing, File Systems, On-Demand Delivery, P2P, Swarming, Virtual Appliances, Virtual Machines

1. INTRODUCTION

Virtual Appliances (VAs) are Virtual Machines (VMs) geared towards a specific set of tasks. They require little or no configuration, working out-of-the-box. VAs fit neatly into the Cloud Computing paradigm - many copies of an identical machine can be launched in a data center, or home/business users can grab the appliance they need from the cloud to run locally just for so long as required. Companies and projects whose sole offerings are VAs ready for either desktop or data center use [3, 11] attest to the growing popularity of VAs. VMware's Appliance directory alone currently lists over 1400 VAs available for the VMware family of Virtual Machine Monitors (VMMs) [13].

Current VA distribution generally requires download of the complete virtual disk image, only after which the VA can be run. Given that compressed VA sizes run anywhere from several hundred MB to a few GB, there can be significant delays from the time a user decides he/she wants to run a particular VA until the time that VA can be used. These problems are only exacerbated when demand for particular VAs spikes and server bandwidth resources become the distribution bottleneck.

To address these issues, we have built a peer-to-peer solution (VMTorrent) to support quick and scalable launching of VAs on-demand. Our solution is applicable to home, enterprise, and data-center settings and agnostic of the particular VMM used (e.g., VMware Player [14], VirtualBox [12], Kernel Virtual Machine (KVM) [4]). VMTorrent provides architectural support for efficient just-in-time deployment and execution of VAs. When a new guest VM is spawned, the system begins downloading the required disk image from a P2P swarm. VMTorrent aims to fetch disk image blocks (encapsulated in Bittorrent pieces) in time for their use by the VMM. VMTorrent does so by adjusting peers' piece selection strategies to prioritize those pieces that are needed to ensure the least possible user experienced delay. In this, VMTorrent resembles P2P on-demand video streaming [2] which selects pieces from the playback window with higher priority, while still acquiring enough random pieces to ensure sufficient diversity.



1: Overall VMTorrent architecture

However, unlike video streaming, playback of VMs is both less structured (pieces are often not used in order) and less predictable (after boot-up a user may do any of several tasks - each accessing a different sequence of virtual disk image blocks). Moreover, while video streams will generally eventually use all blocks of the stream, we have found that only a minority of VA blocks (e.g., 10%) are needed to support user tasks (e.g., many programs, drivers, and files are never used in the average execution). Consequently, VMTorrent consults a set of schedule profiles associated with each guest VM to determine the optimal streaming order - dependent on the expected execution pattern. Execution patterns will differ from one task (e.g., edit a document) to another (e.g., display a presentation). Moreover, the actual schedule may change dynamically in response to demand for specific blocks from the VMM that are outside of the profile used or that prompt transition from one profile to another that better matches the user's apparent usage pattern. In this way our system facilitates fast and smooth deployment of guest VMs while minimizing stalls due to missing data blocks.

2. ARCHITECTURE

Figure 2 illustrates the high-level operation of the system. The figure depicts a set of hosts that are part of, or attached to a cloud. For instance, a host can be a user's computer

SIGCOMM'10, August 30–September 3, 2010, New Delhi, India.
ACM 978-1-4503-0201-2/10/08.

connected to a cloud, or a data center node inside a cloud. Each host consists of a VMM that can run guest VMs, and an instance of VMTorrent that provides the filesystem services for the VMM. These services include downloading the guest VM data and storing a locally modifiable copy of the disk image. The collection of VMTorrent instances forms a P2P network. Each instance also keeps a pristine copy of the disk image so that it can serve that data to peer VMTorrent instances.

VMTorrent essentially operates as a remote file server, and is fully transparent to the VMM (allowing our system to work with arbitrary VMM software). It provides a standard filesystem view that is indistinguishable from that of other common remote filesystems such as NFS, CIFS, and FUSE [10]. Initially the filesystem is empty. To launch a new guest VM, VMTorrent first creates a suitable placeholder in the filesystem for the disk image. Then, it begins to download the contents of the disk image from the P2P network to fill the local placeholder, while serving disk access requests from the VMM. Pieces are downloaded based on profiles of different execution patterns of that VA, allowing the most important blocks to be fetched/pre-fetched so as to minimize stall time. If a request arrives for a block that is not yet present, the priority of the corresponding piece will be increased and the VMM request will stall until the block is obtained from the swarm.

3. DEMO

Our demo will demonstrate the use of VMTorrent for on-demand delivery of VAs in both client-server and swarm settings, including that of a local flash crowd. Visitors will be provided diagnostic views of both the file server, and P2P client as VMTorrent downloads and runs Virtual Appliances of at least two different flavors (Ubuntu and Win7) in real-time. Visitors will not only be able to see pre-scripted executions run on these VMs but also be able to interact with them, executing tasks of their choice from the full set of software available on these VMs.

4. RELATED WORK

The most similar work to ours is that of [15]. This paper proposes a play-on-demand solution for desktop applications. The basic idea is to store user's data at a USB device, and at run time, download desktop applications using P2P (specifically, via BitTorrent). These applications are then run in a lightweight virtualization environment. The main focus here is on how to run an application without installation. The downloaded images here are not standalone VMs, making this approach of more limited applicability. These images are also significantly smaller than those of VAs, consequently they do not explore execution during on-demand download, or VM profiling.

Similarly [8] proposes the idea of using BitTorrent as fast method to distribute VMs to student machines in a training environment. It provides a proof-of-concept, but also focuses on a niche application space and leverages neither on-demand download nor VM profiling. In the popular media [9] advocates the idea of using BitTorrent to perform large scale deployments of desktop images over the WAN.

Also within this space are the Collective [1], a server-based system delivery of managed desktops to personal computer (PC) users. The Collective stores a portion of the managed desktop in the local cache. While it does not profile the particular blocks needed to support different tasks, the Collective will pre-fetch and fill its cache with the most popular applications. Less popular applications are streamed on-demand. This work evolved into the commercial solution MokaFive [7]. VM fork [6] provides a data-center oriented method for instantaneously cloning a VM into multiple replicas running on different hosts Finally, Internet Suspend/Resume [5] was early work that allowed machines to be suspended on one hardware platform, transferred over the network and resumed on another using virtualization.

5. REFERENCES

[1] R. Chandra, N. Zeldovich, C. Sapuntzakis, and M. S. Lam. The Collective: A Cache-Based System Management Architecture. In *NSDI*, pages 259–272, Berkeley, CA, USA, 2005. USENIX Association.

[2] Y. Huang, T. Z. J. Fu, D. M. Chiu, J. C. S. Lui, and C. Huang. Challenges, Design and Analysis of a Large-scale P2P-VoD System. In *ACM SIGCOMM*, Seattle, WA, USA, August 2008.

[3] Jumpbox. `http://www.jumpbox.com/library`.

[4] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. kvm: The Linux Virtual Machine Monitor. In *Ottawa Linux Symposium*, 2007.

[5] M. Kozuch and M. Satyanarayanan. Internet Suspend/Resume. pages 40–46. IEEE Computer Society, 2002.

[6] H. A. Lagar-Cavilla, J. A. Whitney, A. M. Scannell, P. Patchin, S. M. Rumble, E. de Lara, M. Brudno, and M. Satyanarayanan. SnowFlock: Rapid Virtual Machine Cloning for Cloud Computing. In *EuroSys*, pages 1–12, New York, NY, USA, 2009. ACM.

[7] MokaFive. `http://www.mokafive.com/products/components.php`.

[8] C. M. O'Donnell. Using BitTorrent to Distribute Virtual Machine Images for Classes. In *SIGUCCS*, pages 287–290, 2008.

[9] M. Roth. Using BitTorrent to Solve Omnipresent Deployment Problems. *ThinComputing.net*, 2008.

[10] M. Szeredi. FUSE: File System in Userspace `http://fuse.sourceforge.net/`.

[11] Turnkey Linux. `http://www.turnkeylinux.org`.

[12] VirtualBox. `http://www.virtualbox.org/`.

[13] VMware Appliance Marketplace. `http://www.vmware.com/appliances/directory/`.

[14] VMware Player. `http://www.vmware.com/products/player/`.

[15] Y. Zhang, X. Wang, and L. Hong. Portable Desktop Applications Based on P2P Transportation and Virtualization. In *LISA*, pages 133–144, 2008.