From Cluster to Cloud to Appliance

David J. Malan
School of Engineering and Applied Sciences
Harvard University
Cambridge, Massachusetts, USA
malan@harvard.edu

ABSTRACT

We propose a client-side virtual machine (VM) as an alternative to on-campus clusters and off-campus clouds as a development environment for students in introductory courses. In Fall 2011, we deployed the CS50 Appliance, our own such VM, to 600 students on campus and, in Fall 2012, to 700 students on campus and 140,000 students online. We present in this work the results of that two-year experiment. The appliance itself is available as open source for others to adapt or adopt.

Not only did the appliance enable us to provide students with simpler tools, among them a graphical editor without any latency, it also enabled us to provide more sophisticated tools too, including a web server and database server. Moreover, the appliance ensured that the course's workload no longer required constant Internet access, particularly of students abroad. And the appliance alleviated load on the course's servers, with execution of students' programs now distributed across students' own CPUs. Without the appliance (or more costly clusters or clouds), we certainly could not have accommodated as many as 140,000 students.

But some students' laptops, particularly netbooks, struggled under the appliance's weight. Even though designed to be lean, the appliance, like any VM, still consumes resources, particularly RAM. And unforeseen technical difficulties arose in both years, most, but not all, of which we redressed with mid-semester updates and documentation.

Overall we have judged our deployment of an appliance a success, superior to past years' clusters and clouds. And we continue to refine the appliance for Fall 2013.

Categories and Subject Descriptors

D.2.6 [**SOFTWARE ENGINEERING**]: Programming Environments; K.3.1 [**COMPUTERS AND EDUCATION**]: Computer Uses in Education—*Distance learning*;
K.3.2 [**COMPUTERS AND EDUCATION**]: Computer and Information Science Education—*Computer science education*

ITiCSE'13, July 1–3, 2013, Canterbury, England, UK.

General Terms

Design, Experimentation, Standardization

Keywords

virtual machine, virtualization

1. INTRODUCTION

Computer Science 50 (CS50) is Harvard University's "introduction to the intellectual enterprises of computer science and the art of programming" for majors and non-majors alike, a one-semester amalgam of courses generally known as CS1 and CS2. The course is required of majors, but most of the course's students are non-majors. In Fall 2012, enrollment was 715, 73% of whom had no prior CS experience, 20% of whom had taken one prior course, and 7% of whom had taken two or more. The course is taught mostly in C in a command-line environment, with PHP and SQL introduced toward term's end in the context of web programming. Weekly problem sets (i.e., programming assignments) demand upwards of 15 hours per week of most students.

Some years ago, students wrote most of their code on a load-balanced cluster of Linux servers to which they could connect via SSH in order to use `gcc` and `gdb` to compile and debug. In Fall 2009, we moved the course into the "cloud" [2], whereby we recreated that cluster using virtual machines (VMs) within Amazon's Elastic Compute Cloud [1]. Our goals were both technical and pedagogical. As computer scientists, we wanted more control over our students' environment than the university's shared infrastructure would allow (e.g., root access), so that we ourselves could install software at will and respond to students' needs at any hour without an IT department between us and our systems. As teachers, we wanted easier access to our students' work (as via `su`) as well as the ability to grow and shrink our infrastructure as problem sets' computational requirements demanded. And we also wanted to integrate into the course's own syllabus discussion of cloud computing and, with it, scalability, virtualization, and multi-core processing.

On the whole, our new home in the cloud proved a success, as we achieved precisely the autonomy and conveniences we had sought. Moreover, we no longer had to worry about power or cooling or hardware more generally.

But the experiment was not without downsides as well. Serving as our own system administrators cost us time, as did some (self-induced) technical difficulties with software that, in years past, would have been tackled by the university itself. The cloud-based cluster also suffered from

latency, whereby X applications (e.g., `emacs`) no longer performed as well. And even though the course's staff now had root access, students still did not. The cloud (like the on-campus cluster) thus limited students to only those actions that could be performed in user space, even though the course's final project encouraged students to explore languages and tools beyond those covered in class (and installed by us in the cloud).

We thus decided in Fall 2011 to transition from server-side VMs to client-side VMs, whereby students would effectively run their own copy of the course's environment on their own laptop or desktop by installing the CS50 Appliance, a VM preconfigured by us for precisely that purpose. With so many students (particularly without prior background), we preferred that the course still provide students with a standard environment, rather than expect them to install and configure an assortment of tools. (In another introductory course had that approach yielded far too many technical difficulties.) We hypothesized that students' own CPUs were finally adequate to support client-side virtualization, which tends to be resource-intensive. And we were eager to introduce graphical tools (without latency) beyond those available at a command line, among them the simplest of text editors, `gedit`. Having used `emacs`, `nano`, and `vim` in years past (and having experimented with heavier-weight IDEs like Code::Blocks, Eclipse, and NetBeans), we saw in `gedit`, with its embedded terminal window and syntax highlighting, precisely the balance of simplicity and code-friendly features that we had long wanted in the course's first weeks. We did not want tools and arcane keystrokes to get in the way of actual lessons. At the same time, we wanted students exposed to more sophisticated tools and techniques by term's end. In that this appliance would also allow students to administer their own web server (Apache) and database server (MySQL), and anything else via `sudo`, it offered precisely the "low floor" and "high ceiling" [10] that could accommodate students less comfortable and more comfortable alike.

Moreover, as it turned out, this same appliance would ultimately enable us in Fall 2012 to open the course to thousands of students online via edX [4], without needing to scale an on-campus cluster or off-campus cloud. Indeed, since 2011, we have deployed the CS50 Appliance to more than 1,300 students on campus and 140,000 students online. The appliance itself is available as open source at `http://cs.harvard.edu/malan` for other courses to adapt or adopt.

In the section that follows, we present the CS50 Appliance, its implementation details, and its pedagogical design. In Section 3, we present related work. In Section 4, we present the results of our two-year experiment with client-side virtualization of the course's environment. In Section 5, we propose future work. And in Section 6, we conclude.

2. DESIGN

The CS50 Appliance is a VM that enables students to run Linux within a window on their own computer, whether their computer runs Windows, Mac OS, or Linux itself. The appliance happens to run Fedora (32-bit, for compatibility's sake), a rather bleeding-edge variant of the Red Hat-based operating systems that we once used in our cluster and cloud that enables students to experiment for final projects with the latest packages of software. But the appliance could certainly be implemented with most any flavor of Linux.

The appliance is implemented by way of two files: one text and one binary. The former describes, in a machine-readable format, the appliance's virtual hardware (e.g., its network adapters). The latter contains the appliance's virtual disk, a serialized copy of what otherwise would be an actual hard drive.

In order to run the CS50 Appliance, a student (or teacher) need only download a ZIP of those files and install a hypervisor (otherwise known as a virtual machine monitor), the latter of which reconstitutes a virtual machine out of those files, thereby enabling the student to boot the disk image within a window. In Fall 2011, we recommended that students use VirtualBox [8], a free hypervisor from Oracle for Windows, Mac OS, and Linux alike. Frustrated by some bugs we encountered, though, we instead recommended VMware Player (for Windows and Linux) [15] and VMware Fusion (for Mac OS) [14] in Fall 2012. Both proved more stable, but whereas the former is free, the latter required of the course a paid subscription to a VMware Academic Program (so that students themselves would not have to pay) [13].

The appliance's disk image, meanwhile, is the product of two files: `appliance50.ks`, a "kickstart" file that instructs Fedora's installer (Anaconda) how to begin the appliance's configuration, and `appliance50.rpm`, an "RPM" (otherwise known as a "package") that instructs the installer how to complete the appliance's configuration. The RPM, too, ensures via updates that the appliance remains up-to-date throughout the semester.

Not only does the appliance, once booted, provide a command line (and, with it, tools like `gcc`, `gdb`, `make`, and the like), it also provides Xfce, a lightweight desktop environment (that doesn't require much RAM) reminiscent of Windows and Mac OS, per Figure 1. But the appliance behaves as not only a desktop client but also a server. In particular, the appliance includes software like Apache and MySQL that students utilize for problem sets and final projects toward the term's end. As such, the appliance resembles by design an actual web host. Although not used in the course, the appliance also includes `g++`, `java`, `perl`, `python`, `ruby`, and more. Installing additional software requires but a single command (`yum`).

For international students and keyboards, the appliance includes support for multiple languages and layouts. And it also includes TeamViewer [11], free software that enables the course's staff to see and control students' appliances remotely as needed to help troublesheet (if students consent).

The appliance includes course-specific software as well, including `check50` (which enables students to check their programs' correctness) [3], `style50` (which reformats code), and `submit50` (via which students submit work).

Although the CS50 Appliance has its origins in CS50, it is designed to be used in downstream courses well, including our courses on functional programming, systems programming, software engineering, and operating systems. Indeed, students can now reconfigure the appliance for other courses with single commands (e.g., for CS51, it suffices to run `yum install cs51`).

Documentation for the appliance's creation, installation, and usage is available at `http://cs.harvard.edu/malan`.

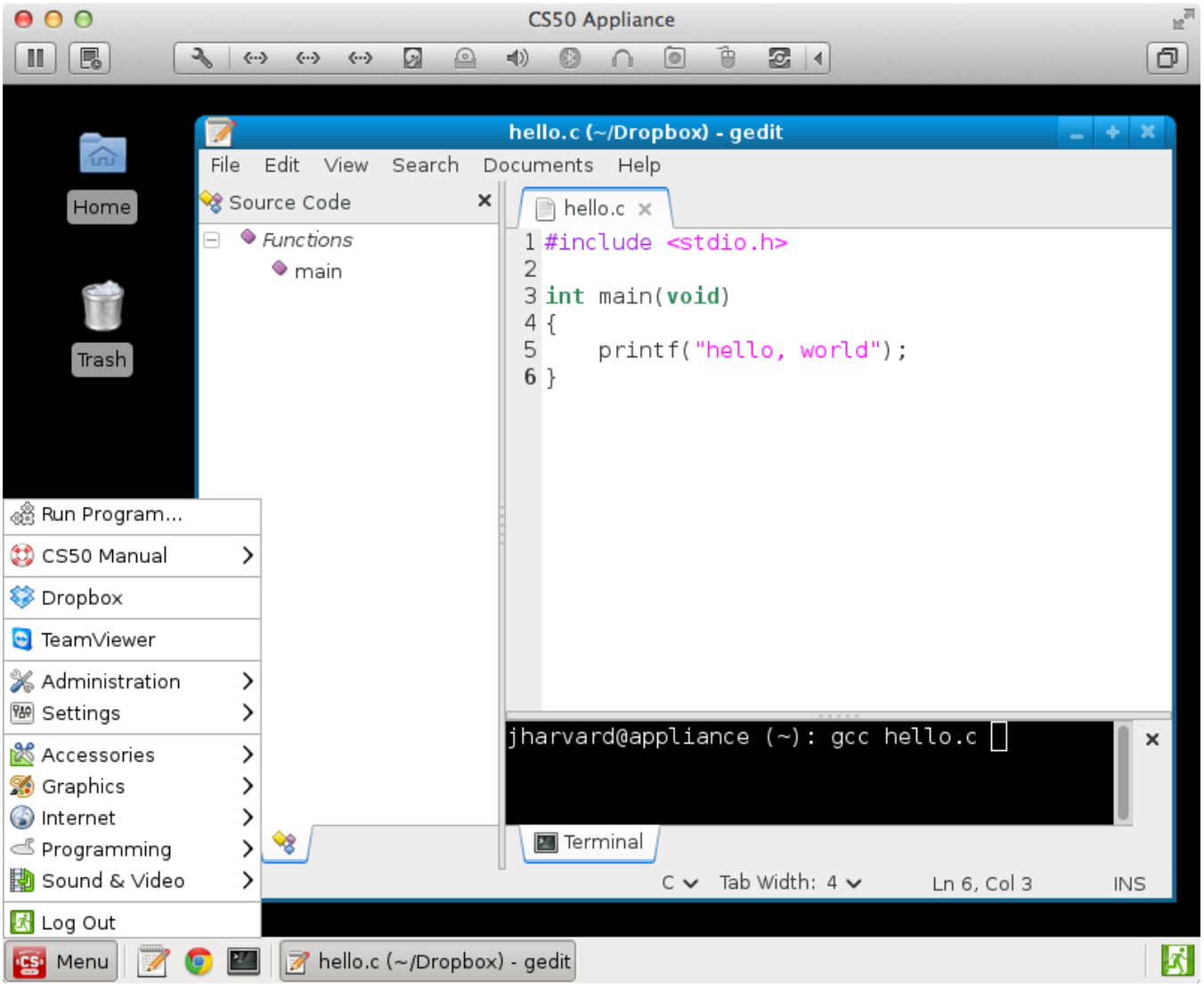

Figure 1: This is the CS50 Appliance, a Fedora-based virtual machine preconfigured for students' use in CS50 (and downstream courses). Depicted here is an open `gedit` window atop the appliance's desktop, with the appliance's menu opened.

3. RELATED WORK

Although courses besides ours have undoubtedly provided their students with preconfigured virtual machines, we have not found any alternatives quite so tailored for use by less-comfortable students in introductory courses. Indeed, most appliances tend to be configured (headlessly no less) with more narrowly defined use cases in mind. We ourselves drew inspiration initially from TurnKey Linux [12], which offers dozens of single-purpose virtual machines for multiple hypervisors. VMware itself offers a "marketplace" of the same. And Amazon similarly offers "Amazon Machine Images" (AMIs), albeit for use in their cloud.

Literature on client-side appliances' use in courses is sparse, but appliances can nonetheless be found in systems courses. Laadan et al. [6] have proposed teaching operating systems using virtual machines, thereby obviating the need for teaching labs. Nieh and Vaill [7], meanwhile, have echoed the same. Griffin and Jourdan [5] have proposed a range of educational use cases for VMs as well.

4. RESULTS

In Fall 2011, we deployed the CS50 Appliance to 607 students on campus, all of whom owned their own laptop or desktop. (Though we also had the appliance installed in computer labs on campus.) All students used the appliance for 8 problem sets (both C- and web-based), and many students used the appliance for final projects (mostly web-based). Overall, the experiment proved a success. Problem sets no longer required Internet access (as they did with the cluster and cloud). Students could more easily use graphical editors like `gedit`. And server load (particularly on nights

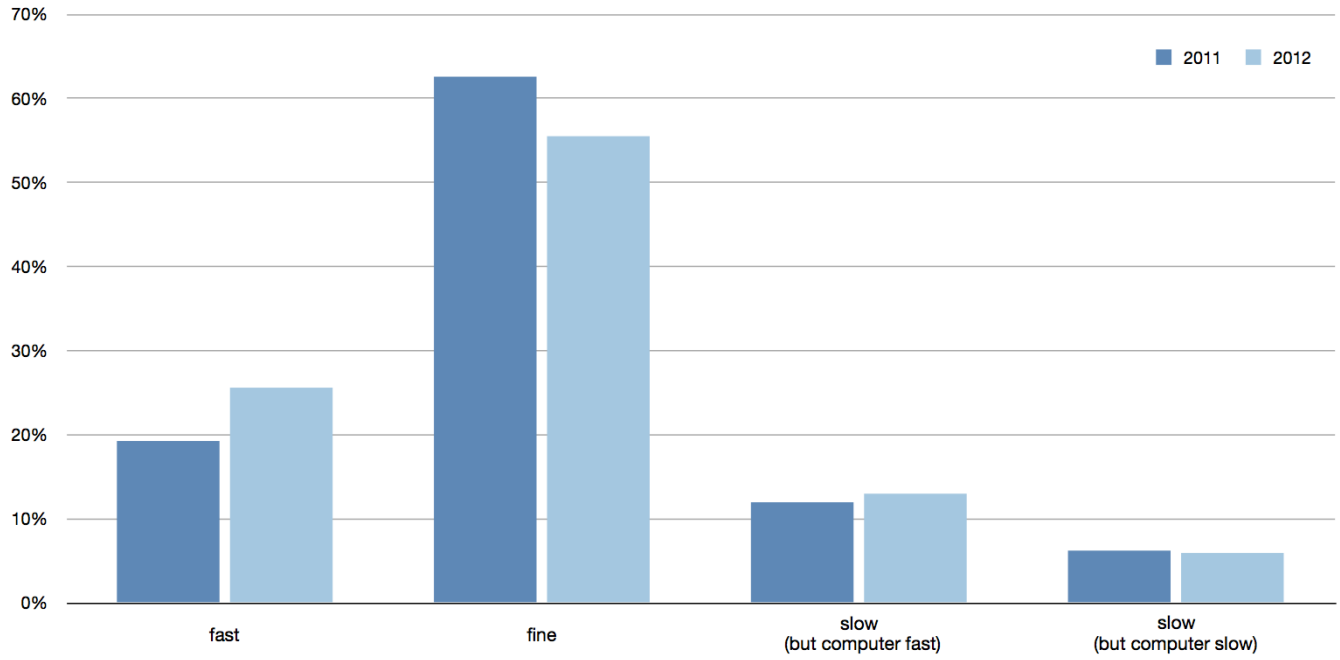

Figure 2: On-campus students' assessments of the CS50 Appliance's performance in Fall 2011 and Fall 2012. Whereas most students described the appliance's performance as "fine" or "fast," a non-trivial percentage of students (just under 20%) felt the appliance was "slow." In some cases, though, students disclaimed that their own computer was also slow.

before deadlines) was no longer an issue, since programs were executed on students' own CPUs.

But the experiment was not without hiccups. Technical difficulties could no longer be resolved server-side for students by staff (though screen-sharing with TeamViewer did help). And some laptops, particularly netbooks, struggled under the weight of a VM. Indeed, per Figure 2, even though most students described (on surveys toward term's end) the appliance's performance on their own computer as "fast" or "fine," just under 20% found the appliance to be "slow," though some students disclaimed that their own computer was slow. Insufficient RAM was often the culprit: nearly 20% of students had computers with only 2GB of RAM, and nearly 3% of students had even less RAM. Older hardware, too, was sometimes to blame. (Though enabling "hardware virtualization" in some computers' BIOSes did sometimes help.) Just over 6% of students had computers that were at least 4 years old.

We encountered unforeseen technical difficulties as well, particularly at term's start. This paper's author made several mistakes in the appliance's configuration, though mid-semester updates (via `yum`) allowed us to fix. Some bugs in Fedora itself also caused us some angst, but we knew we might pay that price by choosing an operating system with a rapid release cycle. Less expected were bugs in VirtualBox, our recommended hypervisor. Corruption of students' virtual disks was surprisingly frequent. We ultimately identified as a trigger students' tendency to close laptops' lids with the appliance still running. (VMware Player and Fusion tolerate such reasonable behavior much better.)

By Fall 2011's end, though, we had redressed most of these problems, either through technological fixes or documentation. We were thus comfortable deploying the appliance again in Fall 2012, this time to 715 students on campus and over 140,000 online, albeit with some modifications. We configured the appliance to use Dropbox (free file-sharing software, up to 2GB) to store students' work (if students consented) so that each student's appliance would be effectively disposable. Even if a student's appliance failed in the midst of some problem set, the student could simply reconstitute another, resync their files, and resume their work within minutes. (The appliance also uses `rsnapshot` internally to back up students' files locally every few minutes.) We also began to recommend VMware Player and Fusion over VirtualBox, in light of the latter's tendency to corrupt virtual disks, despite our misgivings with Fusion's cost. (For edX, we arranged for free 6-month licenses for all students.)

To be fair, even Fall 2012's deployment not without downsides. Per Figure 2, a non-trivial percentage of students on campus (again just under 20%) still felt the appliance was slow. For those students, SSH connections from (less resource-intensive) terminal windows to yesteryear's cluster and cloud would have felt more responsive. As expected, some off-cycle updates to Fedora itself did trigger some unforeseen headaches. And, among our thousands of students online, we probably witnessed every possible technical difficulty. But the course's documentation and forums often helped in those cases.

5. FUTURE WORK

Even so, the CS50 Appliance is a perpetual work in progress, particularly as a new version of Fedora is released each year. To be sure, we could certainly settle on one version of Fedora for multiple years, but we prefer to provide students with the latest versions of software.

On our short-term horizon is to integrate Puppet [9], open-source software for configuring systems, which should help us ensure that mid-semester updates to Fedora itself don't conflict with our own customizations. We may also create a 64-bit version of the appliance so that its architecture is identical to some servers that the course still maintains on campus and off.

For students for whom the appliance's performance remains an issue, we intend to create an AMI within Amazon's Elastic Compute Cloud so that those students can boot an appliance in the cloud, otherwise identical to one running locally, to whose desktop they can connect via a local thin client (as via FreeNX or VNC).

And yet, longer term, for some problem sets, we aspire to replace the appliance altogether with a web-based code editor and debugger that we have begun to develop [3].

6. CONCLUSION

Despite suboptimal performance on some students' laptops, the CS50 Appliance has thus far proved a success overall, a welcome replacement for prior years' clusters and clouds. Not only has the CS50 Appliance enabled us to provide students with precisely the environment we have wanted, it has proved useful beyond the course's own term. On campus, the appliance is now used in several downstream courses as well, and some students even use it for projects of their own. It is, after all, designed to be a development environment, tailored for classroom use but no less versatile for it. And we expect that Moore's Law will help us redress those lingering issues of performance, particularly as seniors with old laptops graduate and freshmen with new laptops enroll.

In the meantime, we intend to redress what issues remain, even as we look ahead toward some future term in which a web-based environment might replace the appliance for some problem sets. Particularly in the course's first weeks, when so much is new to so many students, we would prefer to avoid altogether potential client-side difficulties that might otherwise distract from those weeks' lessons. But, inasmuch as the appliance encapsulates a more real-world environment, albeit prepackaged, we would still introduce the appliance some weeks into the course, at which point students would have all the more background and comfort to troubleshoot client-side difficulties that might still arise.

7. ACKNOWLEDGMENTS

Many thanks to Glenn Holloway of Harvard for his assistance with this work and to our friends at VMware for their support of this work.

8. REFERENCES

[1] Amazon Web Services. Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2/.

[2] David J. Malan. Moving CS50 into the Cloud. In *15th Annual Conference of the Northeast Region of the Consortium for Computing Sciences in Colleges*, Hartford, Connecticut, April 2010.

[3] David J. Malan. CS50 Sandbox: Secure Execution of Untrusted Code. In *44th Technical Symposium on Computer Science Education*, Denver, Colorado, March 2013.

[4] edX. https://www.edx.org/.

[5] T. F. Griffin, III and Z. Jourdan. Educational Use Cases for Virtual Machines. In *Proceedings of the 50th Annual Southeast Regional Conference*, ACM-SE '12, pages 365–366, New York, NY, USA, 2012. ACM.

[6] O. Laadan, J. Nieh, and N. Viennot. Teaching Operating Systems Using Virtual Appliances and Distributed Version Control. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, SIGCSE '10, pages 480–484, New York, NY, USA, 2010. ACM.

[7] J. Nieh and C. Vaill. Experiences Teaching Operating Systems Using Virtual Platforms and Linux. *SIGOPS Oper. Syst. Rev.*, 40(2):100–104, Apr. 2006.

[8] Oracle. VirtualBox. https://www.virtualbox.org/.

[9] Puppet Labs. Puppet. http://info.puppetlabs.com/download-puppet-open-source.

[10] Seymour Papert. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, 1980.

[11] TeamViewer GmbH. TeamViewer. http://www.teamviewer.com/.

[12] TurnKey Linux. Virtual Appliances. http://www.turnkeylinux.org/.

[13] VMware, Inc. VMware Academic Program. http://www.vmware.com/partners/academic/.

[14] VMware, Inc. VMware Fusion. http://www.vmware.com/products/fusion/.

[15] VMware, Inc. VMware Player. http://www.vmware.com/products/player/.