Journal of Digital Imaging

DCMTB: A Virtual Appliance DICOM Toolbox

Steve Langer,[1] Nick Charboneau,[1] and Todd French[1]

Medical Imaging has been fortunate to see an avalanche of free and open source software become available in the last several years. Applications have been written to enable image viewing/storage/analysis/processing, DICOM and HL7 message parsing, results aggregation, anonymization, and more. While robust, many of these packages are difficult to install and configure. Our group desired an approach that would mitigate the efforts required to use these packages across different projects. We found such a solution in the context of using virtual machines.

KEY WORDS: DICOM, HL7, virtualization

BACKGROUND

Medical imaging and informatics has almost an embarrassment of riches in the form of Free and Open Source Software (FOSS). There are several World Wide Web resources dedicated just to tracking these efforts [1–5]:

1. SourceForge
2. DebianMed
3. LinuxMedNews
4. IDoImaging
5. Wikipedia

A search on the word "DICOM" on SourceForge found 83 projects as of March 2009. The FOSS medical packages listed on Wikipedia span an enormous gamut:

(a) bio-surveillance
(b) electronic medical records
(c) bio-informatics
(d) image processing toolkits
(e) image viewing
(f) image storage
(g) data translation and parsing (i.e., HL7 to DICOM or XML)

There are an abundance of tools with many being quite demanding to install, configure, and test. In a recent installation of the popular tool DCM4CHEE, our group invested approximately 1 person-week to successfully install the supporting database, Java server engine, and get all parts working together correctly. [To be fair, part of this was due to the added complexity of having other Java server applications installed which consumed some of the default resources that DCM4CHEE desired.] Our group has about ten members. At any time, there are often several team members working on different projects. If all shared the same instance of a "toolbox", members could overwrite each other's efforts. What is desirable is to have a toolbox that would:

(a) provide a collection of image processing, handling, programming tools, etc.
(b) be owned by each team member
(c) could be carried from project to project.

Other practitioners have arrived at similar conclusions. The maintainers of DebianMed seek to provide a Debian Linux based distribution that has packages designed around specific tasks [6]. This is a remarkable achievement and is very adequate for many uses, particularly for developers who largely sit at a fixed workstation everyday and can count on no one else altering their configuration. However, DebianMed still fell short of our need to have a "toolbox" that could be

[1]From the Mayo Clinic, Rochester, MN, USA.

Correspondence to: Steve Langer, Mayo Clinic, Rochester, MN, USA; e-mail: langer.steve@mayo.edu

Copyright © 2009 by Society for Imaging Informatics in Medicine

Online publication 25 August 2009

doi: 10.1007/s10278-009-9230-8

carried by team members from project to project. Furthermore, there were some tools that did not have a DebianMed package.

VIRTUAL MACHINES

The rise of virtual computing is becoming a well-established trend in computer data centers. Administrators appreciate the ability to run several "virtual" computers on one real physical server. This allows more complete utilization of powerful servers, reduces physical space requirements, as well as power and cooling needs. But an additional benefit is portability. Virtual machines can literally be shut down, packaged, carried across town or country, and installed elsewhere without modification. Virtual computers are just a collection of files which enables portability.

To understand the implications of virtual computers, it is helpful to follow the process of one's creation. A physical computer consists of real hardware (processor, memory, video, networking, etc.) over which is layered software: device drivers, an operating system (OS), and user applications. A virtual computer is similar, except it exists inside a host application on top of another OS. The host application *contains* the virtual machine (VM) and the virtualized OS and its device drivers interact with the host application's virtual hardware. The disks, video, networking, and other services in the hosting "container" application are really software imitations of real devices. However, the VM never knows this, and proceeds as usual. There are several key advantages to this.

Freedom from Fixed Positions and Broken Servers
Since the VM is just a collection of files, it can be packaged and moved to any other physical computer that can support the virtual host application. This promotes both great reliability and portability. If the host physical server breaks down for any reason (and assuming the VM is backed up to external media, like a DVD or flash drive), a user can just start the VM on a totally different server. It does not matter if the hardware is completely different; the VM sees only the virtual hardware of its hosting container application. These features lead to …

Freedom from Conflicting Users
A fixed server installation can be made arbitrarily complex and full featured, as is demonstrated by the DebianMed distribution. However, a large team can encourage conflicts if the individuals are testing different equipment. For example, consider a developer working on an image pre-processor that would sit between a modality and an image archive. If others are attempting to use that same pre-processor while the first developer is altering the code, the opportunity for conflicts is high. Numerous similar examples could be noted. By using a VM, each team member can in principle have their own server, or "toolbox".

Freedom from Complex Physical Server Configurations and Backups
Since the physical servers now need only support the host OS and virtual hosting applications, it becomes practically unnecessary to backup the physical servers. There is no point since the vendors supply DVDs of the host OS and the VM host application. This allows all the complexity of application management and data backups to be moved to the VMs which are easily backed up via simple file copies to external media.

Freedom from Hardware Obsolescence
Often one finds that newer hardware lacks support for legacy OS. This can be particularly painful if one has certain legacy applications that require a certain OS to run on and for whatever reason cannot be upgraded (the vendor went out of business, etc). A VM host offers a consistent set of virtual device drivers across all OS, hiding the newer unsupported hardware, and thus can offer the legacy OS an environment to run in.

Freedom from OS Obsolescence
In another instance, one finds that certain applications run only under a certain version of an OS. Newer OS versions can drop required functions and/or libraries. And yet, the vendor may cease to offer support (bug patches and security updates) for the older OS which make it increasingly risky, and possibly untenable, to continue to run. Virtual computing offers an alternative in the following way. A VM can "hide" its network presence behind the host machine's internet address. This allows the host to filter network packets with current virus

scanners and other protection. In addition, if the VM still gets compromised for any reason, it can be rolled back to an earlier version before the compromise. This is possible because the host application can take “snapshots” of the VM at any phase of its existence by simply copying a then current set of VM files to a saved folder.

METHODS

There are several key questions to be resolved when constructing a VM:

- OS to be virtualized
- Virtual hosting software to use

Further considerations for this project (which actually impacts the prior questions) are, “What applications and tools sets are needed?” Question two was easily answered at our site. We chose VMWare Workstation because it is a standard in our data centers and has a convenient method for packaging VMs for redistribution (VMWare, an EMC Company; EMC, Hopkinton MA, USA). An added benefit is that the free (but not open source) VMWare Player can be accessed from the VMWare web site and used to run the appliance on Windows (Microsoft Corp., Redmond WA, USA), Macintosh (Apple Computers, Cupertino CA, USA) and Linux host platforms. One may ask if there is not a better FOSS choice for the VM platform. This point is complex—both of the leading virtualization vendors started as open source projects but were bought; VMWare by EMC (http://www.emc.com/products/family/vmware-family.htm) and XenSource by Citrix (Citrix Systems Inc., Fort Lauderdale FL, USA, http://citrix.com/lang/English/home.asp). However, since both projects were originally released as open source, both vendors are still legally required to permit distributions based on the original open source modules: for VMWare see http://www.vmware.com/download/open_source.html and for Xen see http://xen.org. There are now other contenders as well. Sun offers an open source VM platform called VirtualBox (http://www.virtualbox.org). In any case, a useful feature of both Xen and VirtualBox is their ability to read VMware image files. Hence, a potential DCMTB user need not be concerned about the “out of the box” VMWare configuration, but can simply import the appliance into their preferred VM host.

The choice of target OS for the toolbox was arrived at after two considerations: the ability to freely redistribute the VM and the primary packages that were required. The former requirement leads immediately to considering FOSS tools, and the bulk of these are available on Linux. While some are also available for Microsoft Windows, the Windows OS itself is not free and cannot be redistributed legally (Microsoft). Hence, a FOSS OS is required. An obvious candidate that we have alluded to is DebianMed. However, as it turned out, some required packages were not available on Debian, and CentOS was chosen instead Centos [7]. A final question was whether to use a 32-bit versus 64-bit version of the OS; we chose 64-bit. While this could be problematic to some potential users with hardware made before 2006, we considered the risk to be minimal considering that imaging professionals will usually have powerful computers.

Having identified our OS and VM platform, the time arrives to construct the VM. There are a multitude of choices to be made at this time, but they can be classed into two main categories: choices that are irreversible after the VM is made, and those that can be changed later when the VM is running and are thus less critical.

Fixed Options

Since these are irreversible choices for the particular VM being created it is worth some time to consider them. As most modern PC architectures have 64-bit capable CPUs (even if the host OS is only 32-bit), we chose 64-bit CentOS. We also created two primary disks: a 32 GB system disk (to provide growth for applications) and a second 8 GB disk for user accounts. The accounts disk can be updated as needed as user needs on a particular instance of the VM grow.

- OS Choice and Version: Windows, Linux, Solaris, Novell Netware. Thirty-two and 64-bit options.
- VM Name: Actually the VM name can be changed later, but changing the directory and file names (which inherit that name) is very difficult without causing failures.
- OS Disk Type and Size: SCSI vs. IDE
- Video emulation: set automatically

Changeable Options

- Number of CPUs: one or two
- RAM: minimum of 1 GB
- Peripherals: CD, USB, Sound Card, floppy disk, added disks
- Network Type: Proxy or direct exposure to internet

Ultimately, our VM was provisioned with the following resources:

- OS: CentOS V5.2 64-bit
- CPU: one
- RAM: 2 GB
- OS Disk: IDE with a 32 GB size
- User Disk: IDE with 8 GB
- Networking: Direct to internet
- Peripherals: CD, floppy, USB support and default video and sound

Once the VM has been created and resourced, the time comes to actually install the OS. This is done simply by using an image file of the OS installation CDs or DVD, and pointing the virtual CD player to that disk image. We used the CentOS 5.2 install image, and proceeded within the VM to install CentOS in the usual way. In addition to the normal Linux services one would install, we also included: sendmail, Apache web server, Windows, and NFS (Network File System) file sharing and the PostGresSQL database PostGresSQL [8]. These services form the cornerstone of the tools we would later layer on the VM.

Programming Tools

The primary purpose of this VM was to provide a set of tools for our HL7/DICOM programming team. As such, an early requirement was a programming environment. We installed the Eclipse platform integrated development environment Eclipse [9]. In addition to the base Eclipse environment, we have installed support for various languages and their associated web development frameworks: Java, C and C++, ruby/Rails, groovy/Grails, and python/Django. Additional languages are also available without explicit Eclipse support (TCL, jython, PERL, etc.)

Web Tools

The following applications represent enormous achievements in the FOSS arsenal of healthcare software. However, they are not at all trivial to install.

DCM4CHEE:
This is a collection of software utilities that together act as an IHE (Integrating the Healthcare Enterprise) compliant Image Archive/Manager cite DCM4CHEE [10]. It requires the use of the JBOSS java application server engine to provide its web interface and communicate with the backend database (PostGresSQL in this case).
MIRTH:
This software collection allows integration with both HL7 and DICOM data streams cite MIRTH [11]. The associated rules engine provides flexible workflow adaptations and mappings based on the inbound data type. It requires a backend database for its operations (PostGresSQL in this case).
XNAT:
The Extensible Neuroimaging Archive Toolkit is the product of a multi-university consortium to develop collaborative archive and image processing tools for neuro-imaging researchers XNAT [12]. It requires the Apache-Tomcat java application server engine to provide its web interface and communicate with the backend database (PostGresSQL in this case).
Diagnostic Medical Physics Tools:
This is a collection of algorithms and software by one of the authors to assist diagnostic medical physicists performing modality quality assurance and imaging network maintenance Langer [13–15].

Command line Tools

DCMTK:
This is a collection of C/C++ libraries and applications implementing large parts of the DICOM standard. It includes software for examining, constructing, and converting DICOM image files, handling offline media, sending, and receiving images over a network connection DCMTK [16].
DCM4CHE2:
This is a library similar in concept and scope of functionality to DCMTK, but implemented in java DCM4CHE2 [17].

Tudor DICOM_Viewer:
A simple viewer based on the DCM4CHE2 toolkit and some code from the ImageJ project Tudor [18].
ImageJ:
A lightweight java based DICOM viewer sponsored by the National Institutes of Health ImageJ [19].
Insight Toolkit:
A C/C++ library to perform image segmentation and registration, sponsored by the National Library of Medicine Insight [20].

An excellent question which may be raised at this point is the issue of preserving the work one has invested in the appliance when it is upgraded. This is not a issue unique to VMs however, whenever one upgrades a Linux distribution there are many risks; inconsistent library versions resulting in broken dependencies, package updates that overwrite user configurations and data, etc. A key design choice in DCMTB should assist in managing this risk; namely the isolation of the user directories to a separate disk partition. By performing system upgrades on the system partition only, a user can download the latest version and simply copy over the user partition from the old VM to the new one. This preserves the user's personal settings, documents, and source code projects. Hence, the only risks during an upgrade are loss of system configurations or data stored in the PostGresSQL database that are on the system partition. For database preservation, we use the "pg_dump"

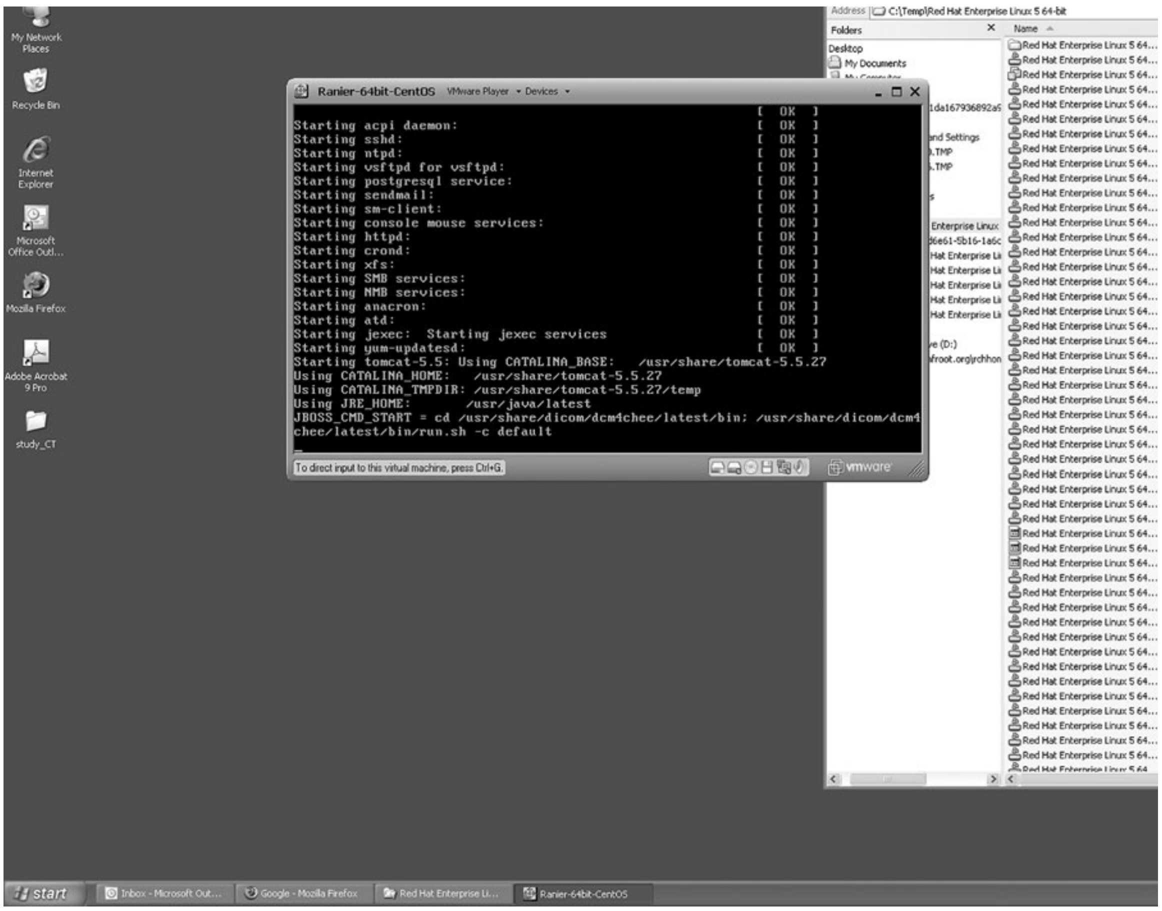

```
                                                          [  OK  ]
Starting acpi daemon:                                     [  OK  ]
Starting sshd:                                            [  OK  ]
Starting ntpd:                                            [  OK  ]
Starting vsftpd for vsftpd:                               [  OK  ]
Starting postgresql service:                              [  OK  ]
Starting sendmail:                                        [  OK  ]
Starting sm-client:                                       [  OK  ]
Starting console mouse services:                          [  OK  ]
Starting httpd:                                           [  OK  ]
Starting crond:                                           [  OK  ]
Starting xfs:                                             [  OK  ]
Starting SMB services:                                    [  OK  ]
Starting NMB services:                                    [  OK  ]
Starting anacron:                                         [  OK  ]
Starting atd:                                             [  OK  ]
Starting jexec:  Starting jexec services                  [  OK  ]
Starting yum-updatesd:                                    [  OK  ]
Starting tomcat-5.5: Using CATALINA_BASE:   /usr/share/tomcat-5.5.27
Using CATALINA_HOME:   /usr/share/tomcat-5.5.27
Using CATALINA_TMPDIR: /usr/share/tomcat-5.5.27/temp
Using JRE_HOME:       /usr/java/latest
JBOSS_CMD_START = cd /usr/share/dicom/dcm4chee/latest/bin; /usr/share/dicom/dcm4
chee/latest/bin/run.sh -c default
_
```

Fig. 1. **In this figure, the *black box* is the VMWare virtual machine container. One can see the DCMTB VM booting the Linux kernel, including the PostGresSQL server which is used by several imaging applications. Note the Windows XP desktop in the background.**

command to save the state of PostGresSQL to flat files that can be moved to safe media. Finally, and perhaps most compelling, one can save the old VM as a backup. When the upgrade is performed, anything broken can be compared to the older VM that can still be run. Discrepancies can thus be resolved. This option is not easily available on physical systems.

RESULTS

The final build uses approximately 35% of both the OS and user disks. The VM is packaged as a single compressed file that consumes about 5.6 GB. As of Version 1.0, the VM consists of the following items in addition to further database tools and command line programming languages. Installation times are listed in ():

Web applications (all tied to PostGresSQL)

- DCM4CHEE (3 days)
- MIRTH (2 days)
- XNAT (3 days)

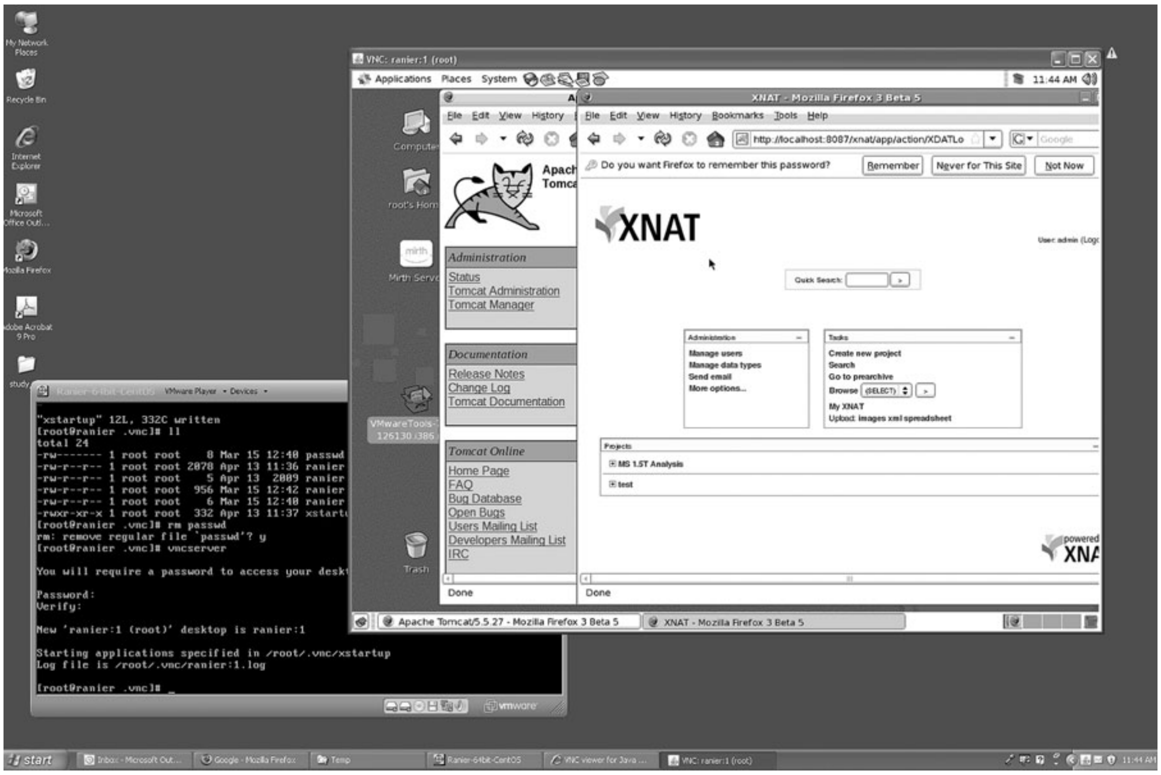
Command Line Toolkits

- Dcmtk (0.5)
- Dcm4che2 (0.5)
- Tudor DICOM Viewer (0.5)
- ImageJ (0.25)
- NIH Insight Toolkit (1.5)

Programming Tools

- Eclipse Platform (1.5)
- Java (0.5)
- C/C++ (0.25)
- Python and Pydev (0.5)
- Ruby/Rails (1)
- Groovy/Grails (1)
- PostGresSQL (2)
- PGPAdmin (0.5)
- PGAccess (0.5)

The VM has been tested on both Windows and Linux hosts using VMWare Player (op cit



```
"xstartup" 12L, 332C written
[root@ranier .vnc]# ll
total 24
-rw------- 1 root root    8 Mar 15 12:40 passwd
-rw-r--r-- 1 root root 2078 Apr 13 11:36 ranier
-rw-r--r-- 1 root root    5 Apr 13  2009 ranier
-rw-r--r-- 1 root root  956 Mar 15 12:42 ranier
-rw-r--r-- 1 root root    6 Mar 15 12:40 ranier
-rwxr-xr-x 1 root root  332 Apr 13 11:37 xstartu
[root@ranier .vnc]# rm passwd
rm: remove regular file `passwd'? y
[root@ranier .vnc]# vncserver

You will require a password to access your deskt

Password:
Verify:

New 'ranier:1 (root)' desktop is ranier:1

Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/ranier:1.log

[root@ranier .vnc]# _
```

Fig. 2. In this figure, one of the authors has logged into the VM and started a X-Windows session viewer. The viewer is seen on the *right*, and one can see the usual CentOS desktop menus. Also visible, the Apache-Tomcat Java server, and an XNAT session is available with two projects already defined.

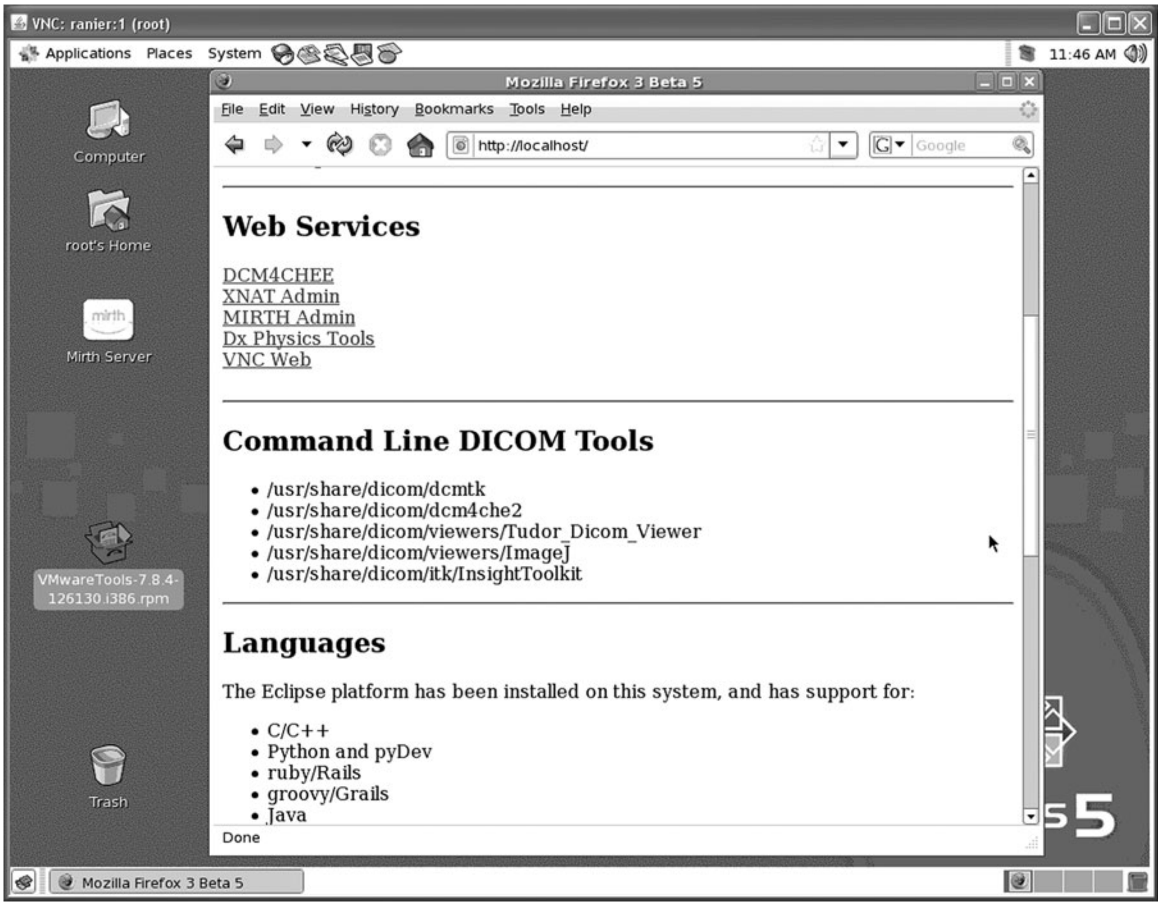
VMWare). On computers using AMD Central Processing Units (CPUs), the host system must have an Athlon-64 Revision D or newer processor, or Opteron/Turion processors of Revision E or newer (AMD, Sunnyvale CA, USA). Intel-based host computers must have CPUs with VT support (Intel, San Mateo CA, USA). The following figures illustrate the VM running on a Windows-based host computer (Figs. 1, 2, and 3).

An example of the toolkit in use is illustrative. One of the authors is involved with a data mining project on DICOM images. The goals were to send images to an archive, forward them to a DICOM parser, copy certain tags to a database, write queries against those tags, and output the results to a web page. Last year, this would have required:

(a) procuring a computer
(b) loading CentOS on it
(c) loading PostGresSQL
(d) loading DCM4CHEE
(e) loading a Web server
(f) loading programming tools
(g) loading MIRTH
(h) configuring MIRTH to use PostGresSQL as the data store
(i) designing a MIRTH channel to harvest the proper tags
(j) testing the combined pipeline of DICOMReceiveràParseràDataBase
(k) writing queries on the database and
(l) publishing the results to the web site

The aforementioned process required about 2 person-weeks to get to step "k". This year, the project required:

(a) locating an available computer
(b) installing VMWare Player on it
(c) loading DCMTB



Web Services

DCM4CHEE
XNAT Admin
MIRTH Admin
Dx Physics Tools
VNC Web

Command Line DICOM Tools

- /usr/share/dicom/dcmtk
- /usr/share/dicom/dcm4che2
- /usr/share/dicom/viewers/Tudor_Dicom_Viewer
- /usr/share/dicom/viewers/ImageJ
- /usr/share/dicom/itk/InsightToolkit

Languages

The Eclipse platform has been installed on this system, and has support for:

- C/C++
- Python and pyDev
- ruby/Rails
- groovy/Grails
- Java

Fig. 3. In this close up of the X-Windows viewer, one can see the details of the CentOS desktop and the VM's main web page with its "table of contents".

(d) writing queries (a preexisting channel was used) and
(e) publishing the output

This year, it required about 2 person-hours to get to the query writing step.

CONCLUSIONS

In the FOSS community, one often hears from a developer that they created a project to "scratch an itch". That is to say they had a need. Because others had shared their previous programming efforts, there were preexisting tools that enabled the current project to be built in a reasonable amount of time.

DCMTB did and continues to scratch an itch for us. As it is used by our team, more tools get thrown into the box, and the opportunities to build meta-tools that span the functionality and database schemas in PostGresSQL present themselves; opportunities that frankly one could not foresee prior to the fusion of the tool sets.

REFERENCES

1. SourceForge http://sourceforge.net/softwaremap Last viewed April 2009

2. DebianMed http://www.debian.org/devel/debian-med Last viewed April 2009

3. LinuxMedNews http://linuxmednews.com Last viewed April 2009

4. IDoImaging. http://www.idoimaging.com/index.shtml Last viewed April 2009

5. Wikipedia http://en.wikipedia.org/wiki/List_of_open_source_healthcare_software Last viewed April 2009

6. DebianMed Tasks http://debian-med.alioth.debian.org/tasks/ Last viewed April 2009

7. CentOS. http://www.centos.org/ Last viewed April 2009

8. Worsley JC, Drake JD: Practical PostGresSQL, 1st edition. Sevastopol: O'Reilly and Associates, 2002

9. Holzner S: Eclipse, 1st edition. Sevastopol: O'Reilly and Associates, 2004

10. DCM4CHEE http://www.dcm4che.org/confluence/display/ee2/Home Last viewed April 2009

11. MIRTH http://www.mirthproject.org Last viewed April 2009

12. XNAT http://xnat.org/ Last viewed April 2009.

13. Langer SG, Kanal K: Spreadsheets for automated data collection, analysis and report generation for Diagnostic Medical Physics. J Digit Imaging 15(2):98–105, 2002

14. Langer SG: OpenRIMS: an open architecture radiology informatics management system. J Digit Imaging 15(2):91–97, 2002

15. Langer SG, Wang J: A goal based cost–benefit analysis for film vs. filmless radiology departments. J Digit Imaging 9 (3):104–112, 1996

16. DCMTK http://dcmtk.org/dcmtk.php.en

17. DCM4CHE2 http://www.dcm4che.org/confluence/display/d2/dcm4che2+DICOM+Toolkit

18. Tudor http://imagejdocu.tudor.lu/doku.php?id=plugin

19. National Institutes of Health "ImageJ". http://rsbweb.nih.gov/ij/

20. National Library of Medicine "Insight Toolkit". http://www.itk.org/