Automated Online Grading for Virtual Machine-based Systems Administration Courses

Lewis Baumstark
Dept. of Computer Science
University of West Georgia
Carrollton, GA 30118
1 (678) 839-6663
lewisb@westga.edu

Edwin Rudolph
Dept. of Computer Science
University of West Georgia
Carrollton, GA 30118
1 (678) 839-6650
erudolph@westga.edu

ABSTRACT
We present a system for automatically and iteratively grading student work in a Systems Administration course. This system can grade – and give feedback regarding – live (running) virtual machines the students have configured. It is appropriate for both face-to-face and online course offerings.

Categories and Subject Descriptors
K.3.1 [**Computers and Education**]: Computer Uses in Education – *Computer-managed instruction, Distance learning.*

K.3.2 [**Computers and Education**]: Computer and Information Science Education – *Computer science education*.

General Terms
Design, Human Factors

Keywords
Online Grading; Automated Grading; Virtual Machines; Systems Administration; Networking

1. INTRODUCTION
The Department of Computer Science at the University of West Georgia offers two system and network administration courses, one each at the undergraduate and graduate level. Both are intended as a blend of theory (e.g., TCP/IP protocol stacks) with hands-on lab work (e.g., configuring DNS and DHCP servers). As others, e.g., [10][8][2] have done, we have transitioned from hardware-based labs full of racks of physical routers and servers to assignments that use virtual machines (VMs) running Linux. One benefit of this approach is our students are no longer required to be physically present in a laboratory to work on their network and server-configuration assignments; instead, they can work on them at home or wherever they have an available Internet connection.

This approach presented logistical problems of its own, however, particularly with respect to grading. Most of our CS courses use the Moodle courseware, which allows assignment and project work to be uploaded for the instructor to grade later. While a virtual machine is nothing more than a collection of files and could theoretically be uploaded, its size (typically in the 5-10 gigabyte range for each assignment) makes that impractical in terms of both network bandwidth and server disk storage. An alternate approach is to require students to demonstrate their work in-person, but this is time-consuming. (Further, our graduate systems administration course is now 100% on-line, making in-person demonstrations impossible.) We have even had students upload screen shot pictures of their configured systems.

One offering of the graduate course was taught using VMware Server. This product allows multiple virtual machines to be stored and run from a single physical server. Each VMs desktop can be accessed remotely using a web-based tool. This arrangement facilitated grading: the instructor could simply log into a student's VM directly and evaluate the quality of their running system. Unfortunately, with eleven students there were serious performance issues (note the undergraduate version of the course usually has 30-40 students, so VMware Server would be even less appropriate). For both the graduate and undergraduate courses we now use VMware Workstation, which allows students to run VMs on their own computers.

With this in mind (students running VMs on their own computers), we desired a grading system with the following qualities:

- *Is modular* in that new assignments can be "plugged-in" to the system.
- *Can evaluate running systems*. For example, test whether a student's DNS server can, in fact, properly resolve host names.
- *Is initiated by the student* when the student is working on the assignment.
- *Can be graded remotely* (e.g., from the student's residence). This means the system must work from behind the typical residential NAT-based router as well as tunnel through our campus firewall.
- *Is asynchronous*. The instructor should not have to be on-line when the work is submitted.
- *Should provide automatic feedback*. The student should receive an immediate score along with some indication of what is incorrect.
- *Allows for an iterative learning process*. Similar to iterative design-implement-test software development cycles, we want these assignments to be completed incrementally, with students working on one part, testing it, and re-working until that part is completed. After completing a step, they can move on to next part of the assignment.

SIGCSE'13, March 6–9, 2013, Denver, Colorado, USA.

- *Secure*. To avoid cheating, each student should be authenticated and the details of their grading sessions should only be visible to the student and the instructor.

2. Related Work

As an alternative to live demonstrations, [10] suggest grading virtual machine assignments by having the students submit a CD-RW disk that has a file diff between the as-assigned VM state and its as-completed state. This means less data transfer than submitting an entire VM file for grading, but it does introduce the overhead in patching a test VM on the grader side, plus the turn-in logistics. It also requires all students start with the same VM for each assignment; our approach allows students who want to experiment with different Linux distributions to do so.

There are several existing approaches for automated and semi-automated grading for programming projects. APOGEE [3] is targeted at web-design projects and uses a specially-designed web client to run testing scripts against student work. Infandango [5] and Web-CAT [1] are examples of systems that accept submitted program code, compile it, and compare the output against expected results.

We were initially inspired by the judging systems such as those used in ACM student programming competitions. The PCSS3 [7] and PC^2 [9] systems use a submit/receive feedback/revise cycle similar to our system; both can be configured to automatically judge submissions.

3. ONLINE GRADING SYSTEM

3.1 System Use Cases

There are three main interactions with the system:

1. **Student solicits feedback**: The student, having completed a part (or the entirety) of an assignment, submits the work to the autograder for review. The system returns a human-readable report with a current grade for the assignment and text giving feedback on correct/incorrect items. (*Note that the grade is not recorded in the course grade book at this time*.) This process takes approximately one minute.
2. **Instructor records grades**: After an assignment is due, the instructor sifts through the records of all grading attempts, by all students, for the assignment, recording each student's highest grade over all their attempts for the assignment.
3. **Student-Instructor conference**: If a student needs instructor help (via email, instant messaging, face-to-face, etc.) beyond the automated feedback for an assignment, the instructor can look at records of that student's previous attempts as a guide to the student's current issues with the assignment. (We have also used this information to aid other tasks such as debugging assignment scripts and network-connection issues.)

3.2 Running Example: DNS Server

In the remainder of this paper we will use as an example an assignment where students configure the BIND domain name system (DNS) server. [6] The requirements are that they:

- Install the BIND package (along with some useful command-line tools).
- Make the DNS server authoritative for the assigned domain.
- Configure the DNS server to resolve host names in that domain.
- Configure the DNS server to forward lookups for hosts not in its own domain to another DNS server.
- Configure the VM's network settings so that it uses itself to resolve host names.

The source code for this and other assignment scripts can be viewed at http://lewisb.wiki.westga.edu/autograder.

3.3 Shell Script Solution

Our system is implemented as a set of Linux shell scripts, written in the bash [4] scripting language. The main script, `grade.sh`, operates as follows: a student (working on their own computer) contacts the grading server requesting it grade, in our example, the `dnsserver` assignment. The assignment is a separate bash shell script, which is transferred to the student's computer and executed locally. After execution of the assignment script, the results are both presented to the student and uploaded to the grading server where they are available for the instructor to view at her convenience. (The assignment script is also deleted from the student's computer once grading is complete.) We strove to make invocation of the grading script as simple as possible; for the example, a student would execute:

```
bash grade.sh dnsserver
```

i.e., just `grade.sh` followed by a parameter indicating the name of the assignment script. This arrangement provides us with modularity; all an instructor need do to create a new assignment is to write a separate, custom assignment script and save it on the server. There is no need to modify `grade.sh`.

By executing a grading script directly on the student's VM, we are able to evaluate the entire state of the running system as a user of that system. Any program that can be run from the command line and whose output can be parsed can be used for evaluation. For example, in `dnsserver`, we want to find out if the student's DNS server has a name server (NS) record defined for their domain, so we execute the command:

```
NS=$(dig ns sysadmin.cs3280 +short)
```

This uses the `dig` command to query the DNS server for a name server that is authoritative for our `sysadmin.cs3280` domain. If the command succeeds (Linux return status 0, available from special bash variable `$?`) we go on to use the value it stored in the `$NS` variable to perform host name resolutions using that server. If it does not succeed, the student receives an informational message describing the error.

Note that there is only a small amount of network data transfer during the grading attempt: the student receives the script file (to date, the largest is 4 kilobytes) and sends back a log of the grading attempt (a few kilobytes or less, depending on the assignment). Compare this to a solution where a student would upload a 5 gigabyte VM file and then wait until an instructor can run and evaluate the system manually.

3.4 The Role of Secure Shell

The `grade.sh` script creates a Secure Shell (`ssh`) tunnel between the student's PC and our grading server. This authenticates the student to the grading server. (Previously, in a one-time setup operation, the student has provided the server with a public key for authentication). It also sets up port-forwarding from the server back to the student's PC, which allows the server to transfer the assignment script (e.g., `dnsserver`) to the student's PC. This step may seem like overkill, but consider that,

Table 1. Results of Previous Course Offerings

Assignment	Description	Average Grade	Average Number of Attempts
`vminstall`	Get a virtual machine installed; also insures the student understands and can perform the auto-grading process.	100	3.1
`static_ip`	Configure a static IP address, including DNS and gateway information, for a server machine	96.4	5.2
`dchp`	Configure a dynamic host configuration protocol (DHCP) server to dynamically assign IP addresses and other configuration information to a DHCP client.	88.3	7.0
`dnsserver`	Configure a domain name system (DNS) server to resolve IP addresses for a domain.	92.5	15.7
`fileserver`	Configure an server message block (SMB) file server to share files across a network	90.8	6.1
`webserver`	Install and configure a webserver hosting static and dynamic content.	92.1	4.2
`firewall`	Configure and write filtering rules for a firewall.	75.0	8.4
`users-and-groups`	Create user accounts and privilege groups.	99.5	2.2
`ldap-server`	Install and configure a lightweight directory access protocol (LDAP) server; populate with directory information.	90.7	4.1
`nat`	Configure a gateway to perform network address translation (NAT) between two private networks.	94.0	3.0
`mrtg`	Configure the MRTG software package, which uses the simple network messaging protocol (SNMP) to display system information as a web page.	86.7	4.0
Unique students: 61 Total attempts: 2642 Number of course offerings: 4 (*not all course offerings used every assignment.*)			

typically, the system being graded is a virtual machine behind two layers of Network Address Translation (NAT): one provided by the VM software itself to define a virtual network on the host machine, the other a student's broadband router defining a private network (e.g., 192.168.1.0/24) behind the single non-private IP address given them by their Internet Service Provider (ISP). The port-forwarding hides this complexity from the grading server, which needs to treat the VM as an `ssh` "server" for purposes of running the assignment script.

3.5 Sample Results File

In Figure 1 (on the last page due to its size), an anonymous student has initiated grading of `dnsserver` and received their results. The results are condensed for brevity and line numbers have been added. These results are copied to the grading server so the instructors can review them.

- Lines 00-01: For troubleshooting purposes, the IP address and the Ethernet adaptor in-use by the VM are reported.
- Lines 03-18: It is useful, again for debugging purposes, to include any relevant configuration files in the results. Shown are the contents of a BIND "zone" file, which contains host and name server records for a DNS zone. The instructor can review these files and advise a student on mistakes they have made. In this example, the student has erred in configuring the NS record for the domain.
- Lines 20-21: These show successful completion of two parts of the assignment (installing the dnsutils and bind9 packages).
- Lines 22-46: These show errors on most of the remaining items of the assignment (setting up an authoritative name server and successfully resolving host names using proper "A" records). Note that the student is told what parts of the assignment that did not succeed.
- Line 47: Part of the assignment is to have the VM look to itself (instead of an external DNS server) to perform its own DNS lookups. The student has configured this part correctly.
- Line 48: The overall score for this attempt.

4. Iterative Completion of Assignments and Instructor Feedback

Students are allowed to attempt these assignments as many times as they wish before the due date. On each attempt, the student sees their current score and results similar to those described in the previous section. Only the grade for the highest-scoring attempt is recorded.

This gives students a feedback cycle for learning as they are not required to "get it right the first time" and can learn from their mistakes and correct them. As each set of results is copied to the instructor, the instructor can use them to guide students through the difficult parts.

Table 1 shows the various assignments made over the (to date) four course offerings where our online grading system has been used. The last column shows that students do make use of the iterative nature of the assignments, bringing them to grades that in most cases are in the A or B range (second-to-last column).

5. Summary and Future Work

Our online grading system allows students in our systems administration courses to engage with hands-on assignments using virtual machines at home but with the benefit of immediate grading and feedback. Further, the nature of the assignments allows for iterative refinement of their assignments, providing feedback along the way.

As this system is capable of remotely grading running virtual machines, it could be adapted to grade a range of student projects that require a system of running software components such as a web design project with a web server, database, and runtime platform such as Java or .NET.

In the future, there are several features we would like to add:

- Scripts to link final grades for an assignment to the student's grade book on Moodle
- Ability to run root-level (administrator) commands on the student's VM. Not having this ability limits some of the analysis we can perform.
- Perform analyses of Internet-based services as a remote client. In other words, we would not just run local commands on a student's machine, but have the ability to make TCP/IP connections to it (for example, acting as an external HTTP client to test a student's webserver configuration).

6. REFERENCES

[1] Allowatt, A. and Edwards, S. 2005. IDE Support for Test-driven Development and Automated Grading in Both Java and C++. *Proceedings of the 2005 OOPSLA workshop on Eclipse technology* (San Diego, CA. Oct. 2005). eclipse'05. 100 – 104. DOI= http://dx.doi.org/10.1145/1117696.1117717

[2] Bullerw, Jr., W.I., Burd, S., and Seazzu, A.F. 2006. Virtual Machines – An Idea Whose Time Has Returned: Application to Network, Security, and Database Courses. *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education* (Houston, TX, USA. March 2006). SIGCSE'06. 102 – 106. DOI= http://dx.doi.org/10.1145/1124706.1121375

[3] Fu, X., Peltsverger, B., Qian, K., Tao, L., and Liu, J. 2008. APOGEE – Automated Project Grading and Instant Feedback System for Web Based Computing. *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education* (Portland, OR, USA. March 2008). SIGCSE'08. 77 – 81. DOI= http://dx.doi.org/10.1145/1352135.1352163

[4] GNU.org. 2010. GNU bash Manual. Retrieved November 13, 2012 from http://www.gnu.org/software/bash/manual/

[5] Hull, M., Powell, D., and Klein, E. 2011. Infandango: Automated Grading for Student Programming. *Proceedings of the 16th Annual Conference on Innovation and Technology in Computer Science Education* (Darmstadt, Germany. June 2011). ITiCSE'11. 330. DOI=http://dx.doi.org/10.1145/1999747.1999841

[6] Internet Systems Consortium. 2012. BIND 9 Administrator Reference Manual. Retreived November 13, 2012 from http://ftp.isc.org/isc/bind9/cur/9.9/doc/arm/Bv9ARM.pdf

[7] Kăcer, M. and Mannová, B., 2008. PCSS3: Programming Contest Scoring System. *Proceedings of Competitive Learning Institute Symposium* (Banff, Alberta Canada. April 2008).

[8] Laadan, O., Nieh, J., and Viennot, N. 2010. Teaching Operating Systems Using Virtual Appliances and Distributed Version Control. *Proceedings of the 41st SIGCSE Technical Symposium on Computer Science Education* (Milwaukee, WI, USA. March 2010). SIGCSE'10. 480 – 484. DOI= http://dx.doi.org/10.1145/1734263.1734427

[9] PC^2 Project. 2011. Programming Contest Control System. Retreived November 13, 2012 from http://www.ecs.csus.edu/pc2/

[10] Volrath, A. and Jenkins, S. 2004. Using Virtual Machines for Teaching System Administration. *Journal of Computing Sciences in Colleges*. 20, 2 (Dec. 2004), 287 – 292.

```
00: You are using ethernet adapter eth0
01: Your IP address is 192.168.244.3
02: ================================================
03: .db/.zone files
04:
05: $TTL 3D
06: @ IN SOA   ns.sysadmin.cs3280. admin.sysadmin.cs3280. (
07:
08:                   2006071801
09:                   28800
10:                   3600
11:                   604800
12:                   38400
13:           );
14: sysadmin.cs3280. IN       NS     ns.sysadmin.cs3280.
15: www.sysadmin.cs3280. IN  NS  ns1.sysadmin.cs3280.
16:
17: www IN     A      192.168.244.3
18: ns1 IN     A      192.168.244.3
19: ================================================
20: dnsutils installed
21: bind9 installed
22: Found k.root-servers.net.
23: e.root-servers.net.
24: d.root-servers.net.
25: i.root-servers.net.
26: l.root-servers.net.
27: c.root-servers.net.
28: b.root-servers.net.
29: m.root-servers.net.
30: j.root-servers.net.
31: f.root-servers.net.
32: g.root-servers.net.
33: h.root-servers.net.
34: a.root-servers.net. (variable QUERYIP) via nameserver
35: Your VM is not authoritative for sysadmin.cs3280
36: Partial score: 50
37: Did not find www.sysadmin.cs3280
38: Partial score: 50
39: Did not find ftp.sysadmin.cs3280
40: Partial score: 50
41: Did not find pop3.sysadmin.cs3280
42: Partial score: 50
43: Did not find ns1.sysadmin.cs3280
44: Partial score: 50
45: Did not find smtp.sysadmin.cs3280
46: Partial score: 50
47: Your VM is set to use itself to resolve host names.
48: Final score: 65
```

Figure 1. Sample Grading Session Results