CSC 510 Advanced Operating Systems, Fall 2017

Dr. Dale E. Parson, Assignment 4, Benchmarking and analyzing a modified Assignment 1 running on System VMs on Type 1 and Type 2 hypervisors. This assignment is due by 11:59 PM on Thursday December 7 via a D2L -> Assessments -> Assignments -> Assignment 4.

This assignment entails no coding. It is strictly reading & analysis. In other analytical assignments I would sometimes accept plausible answers, but for these questions Q1-Q11 below I am looking for exact answers, based on our textbook¹ and in readings linked below.

Perform the following steps after logging into Linux machine mcgonagall.

```bash
    ed $HOME
    mkdir OpSys  # This step should not be needed after assignment 1. It won’t hurt.
    mkdir OpSys/assn4
    cd ./OpSys/assn4
    unzip stepvmInC.assn4.zip
    cd ./stepvmInC
    make test
```

This code base is my solution of assignment 1, enhanced with some operating system function calls to measure the elapsed time and the user and system CPU times of the token-threaded, subroutine-threaded, and direct-threaded implementations of assignment 1’s emulated, stack-based VM written in C. My added functions measure strictly the time spent running the emulated stack-based VM; previous assignments have used the Linux `time` utility, which measures these times for the entire process. Instrumenting the code with time monitor function calls focuses strictly on the run time of the C-emulated VM, without measuring other parts of the processes.

At [http://faculty.kutztown.edu/parson/fall2017/csc510aasn4times.pdf](http://faculty.kutztown.edu/parson/fall2017/csc510aasn4times.pdf) is a schedule of times that each of you may access the two VMs used in this assignment. There is only one account on each VM, with an account name of csc510 and a password of csc510, and there is only one virtual CPU allocated to each VM. **Please follow my instructions below for using the account in order to avoid trashing our running environment; I will charge 20% against the project grade for damaging the account. Also, make sure to follow the schedule for logging into the VMs and running benchmarks. Do NOT run during someone else’s time, and do NOT stay logged in when you are not working.** The benchmarks take less than 5 minutes to run on each VM. You can collect your measurements, and then work on other parts of the assignment without being logged into the VMs. Simultaneous benchmarking by multiple students on a given VM will skew results. We will also be running these benchmarks on mcgonagall, but with 32 hardware processors running up to 8 single-threaded benchmarks, there is not much danger of stepping on each others’ run times.

After running `make test` on mcgonagall as instructed above, review the instructions and emulated VM code for assignment 1’s stack-based machine. Then go on from here.

**How do you collect statistics on mcgonagall, on the Type 1 VM, and the Type 2 VM?**

¹ Our course textbook is available electronically via Rohrbach Library if you don’t own a copy. See our course page for details.
A. mcgonagall

After running `make test` above, run `make testspeed` and record the following results reported by the instrumented timing function calls. Replace “???” with the measured values.

```
/bin/bash -c "time ./stepvmTokenInC 0 BASIC 5000000 > /dev/null"
  run_stepvmInC elapsed real time = ???
  run_stepvmInC user CPU time = ???
  run_stepvmInC system CPU time = ???

/bin/bash -c "time ./stepvmSubroutineInC 0 BASIC 5000000 > /dev/null"
  run_stepvmInC elapsed real time = ???
  run_stepvmInC user CPU time = ???
  run_stepvmInC system CPU time = ???

/bin/bash -c "time ./stepvmDirectInC 0 BASIC 5000000 > /dev/null"
  run_stepvmInC elapsed real time = ???
  run_stepvmInC user CPU time = ???
  run_stepvmInC system CPU time = ???
```

Run `uname -a` and record the results. Run `man uname` to see how to interpret the reported fields. If the reported **hardware platform** contains the string `_64`, the reported machine or VM is running a 64-bit Intel architecture; otherwise, it is running a 32-bit Intel architecture.

Run `lscpu`, then `cat /proc/meminfo | less`, and then `cat /proc/cpuinfo | less`, and record the results. For the latter command record only the output of processor 0; the other 31 processors are identical. The two VMs to be examined later have only 1 virtual processor each. You can probably get by without piping to `less` on the VMs.

B. Linux VM atop a Type 1 hypervisor

From acad or mcgonagall, run `ssh -l csc510 156.12.127.15` and log in with a password of `csc510`. Do this only during your scheduled time slot. This VM runs on the **VMware ESXi Type 1 hypervisor** on machine `kupvxcsit01` as Chris Walck explained in class on November 14. The hardware configuration of the machine `kupvxcsit01` is identical to `mcgonagall` except for the latter’s NVIDIA GPU card used for number crunching; that GPU card does not affect this exercise. We cannot log into this underlying machine because the Type 1 hypervisor that runs on the hardware is not a conventional, time-sharing OS. However, the `cpuinfo` and `meminfo` from mcgonagall in the previous step is identical for this underlying machine.

---

From Chris’ presentation:

![Hardware and processors details](image-url)
Run `ls -l`, and then `cd YOURID` into your subdirectory, which is name of your account on acad as well as your Kutztown ID. Do not stay in the home directory or use anyone else’s directory. Here are the directory names.

awern214  
cccarr419  
dgasp528  
mchel814  
parson  
rnull732  
tcrav850  
zerb789

Next run `unzip ../stepvmInC.assn4.zip`, and then `cd ./stepvmInC`, and then `make test`. Record any diffs that appear in the three .dif files for later analysis.

testbasic.dif  
testdirect.dif  
testsubroutine.dif

Next, run `make testspeed` and record the results as you did for megonagall.

Run `uname –a, lscpu, cat /proc/meminfo | less`, and then `cat /proc/cpuinfo | less`, as you did for megonagall, and record the results.
Log out of the VM when you have finished your work, before the end of your time slot. You may run other Linux utilities while you are in there. Please do not go into other students’ directories or damage the account in any way.

C. Linux VM atop a VirtualBox Type 2 hypervisor running on mcgonagall

From acad or mcgonagall, run `ssh -l csc510 156.12.127.13 (not .15 as before)` and log in with a password of `csc510`. Do this only during your scheduled time slot. Follow all of the data collection instructions in Step B used for the Type 1 hypervisor. Log out during your time slot when done. From Chris’ slides:

Type 2 hypervisor

![Type 2 hypervisor diagram](image)

D. References

There is a lot of reading in this assignment. Start early.

Some questions in the next section refer specifically to topics in Chapters 3 and 8 in our textbook. The textbook is available online via Rohrbach Library. I will go over access on 11/21 in class.


You may skip Section 10.1, but read the remaining sections on the Type 2 VirtualBox VMM. This is a good overview of hypervisor implementation issues. Sections 10.3 through 10.6 are the most important. Note especially the definitions of various instruction-permission **rings** in 10.5. These show up in the Intel and other references on instruction-mode hardware VMM technology.


Slides 6 through 10 of this Intel overview augment the information in the VirtualBox reference above.


This page outlines the information reported by `cat /proc/cpuinfo` in the previous steps.
This page documents the important values for the `flags` field reported by `cat /proc/cpuinfo`. You need some of this information to answer questions. Use only the flags reported for the Intel processors, not for AMD or any others, since we are using Intel in this benchmark.

The diagram and its accompanying text on page 3 of the ESXi Type 1 hypervisor document give a decent overview. Most of the rest of this doc are outside the scope of our project.

E. Questions and Answers (THIS IS WHAT YOU MUST TURN IN VIA D2L)

I am storing a copy of these in file http://faculty.kutztown.edu/parson/fall2017/csc510fall2017assn4.txt. Make a copy of that file, answer the questions in your copy, and turn it into D2L Assignment 4 by the due date. I am looking for precise, accurate answers to the questions, not guesses or plausible answers. I do plan to curve the grade if helpful to you. See the next section below in Grading. Please read this document to the end before you start answering questions.

**Q1:** Chris Walck and I each attempted to set up a VirtualBox Type 2 hypervisor and a VM within it atop the ESXi Type 1 hypervisor on machine kupvxcsit01. Our attempts failed with errors. Given the fact that the VirtualBox hypervisor is a valid 64-bit set of Linux executable processes, but that the ESXi Type 1 hypervisor does not support running it, while the Linux OS on mcgonagall does: What kind of compatibility as defined in Chapter 3 of our textbook does Type 1 ESXi provide? Use precise terminology as used in the textbook, and explain how you arrived at this answer in a brief paragraph.

**Q2:** List these timing results for running assignment 1’s emulated VM (with added timing function calls and print-outs) on mcgonagall and on the Type 1 ESXi VM. Do not include elapsed time this answer.

Compare the results of the two machines. Is one of these machines consistently faster than the other? If so, which one? Explain a reason for any speed differential that you find.

```
/bin/bash -c "time ./stepvmTokenInC 0 BASIC 5000000 > /dev/null"
run_stepvmInC user CPU time = ???:??:??
run_stepvmInC system CPU time = ???:??:??
```

**Q3:** List these timing results for running assignment 1’s emulated VM on mcgonagall and on the Type 2 VirtualBox VM running on mcgonagall. Do not include elapsed time this answer.

Compare the results of the two machines. Is one of these machines consistently faster than the other? If so, which one? Explain a reason for any speed differential that you find.

```
/bin/bash -c "time ./stepvmTokenInC 0 BASIC 5000000 > /dev/null"
```
run_stepvmInC user CPU time = ??? secs
run_stepvmInC system CPU time = ??? secs
/bin/bash -c "time ./stepvmSubroutineInC 0 BASIC 5000000 > /dev/null"
run_stepvmInC user CPU time = ??? secs
run_stepvmInC system CPU time = ??? secs
/bin/bash -c "time ./stepvmDirectInC 0 BASIC 5000000 > /dev/null"
run_stepvmInC user CPU time = ??? secs
run_stepvmInC system CPU time = ??? secs

Q4: Out of the three machines mcgonagall, the Type 1 ESXi VM, and the Type 2 VirtualBox VM, which one or ones **support** hardware virtualization assistance on this machine as explained in the `cat /proc/cpuinfo` output these linked readings? Explain how you arrive at your answer.


Q5: Out of the three machines mcgonagall, the Type 1 ESXi VM, and the Type 2 VirtualBox VM, which one or ones **are** virtual machines as explained in the `cat /proc/cpuinfo` output the Q4 linked readings? Explain how you arrive at your answer, based strictly on the output of interpreting `cat /proc/cpuinfo`.

Q6: Is there anything in the linked documentation for VirtualBox to expand on your answer to Q1 concerning why Chris and I could not set a Virtual Box Type 2 hypervisor atop ESXi? If so, cite the section of this reference and explain.


Q7: Examine any diffs that appear in the three .dif files from running `make test` on the augmented assignment 1 code. Which of the VMs, if either, show one or more diffs? For any VM that shows differences, explain the reason(s) for the diffs. Note that the makefile creates the .dif files as follows.

diff testbasic.out testbasic.ref > testbasic.dif
diff testsubroutine.out testbasic.ref > testsubroutine.dif
diff testdirect.out testbasic.ref > testdirect.dif

The .out files show results of your test runs, and the .ref files show expected output as captured on mcgonagall for assignment 1. I have modified the makefile so that a non-empty .dif does not abort the `make test`. To answer this question you need to understand the program running on assignment 1’s stack-based emulated VM. Consult the user-mode portion of stepvmTokenInC.c that begins at this line.

/* USER PROGRAM starts at location [17], see STEP_LCODEMEM above. */

Q8: Sections 8.2.3 and 8.2.4 of the textbook, which was written in 2005, address the problems of so-called **critical instructions** that should trap to a kernel mode subroutine when executed in user mode, but instead they either work as they would in kernel mode or do nothing (i.e., function as no-ops); such instructions violate the clean separation of kernel mode from user mode, potentially allowing a VM running these instructions to gain access into the underlying VMM (when the instructions execute as they would in kernel mode), or violate the design intent of using them in the VM (when the instructions act as no-ops, having no effects). Section 8.2.4 outlines the approach of scanning and patching the offending
VM code that uses these critical instructions, replacing them with a trap to a kernel mode subroutine that emulates them safely and properly. Can you determine from your linked readings (above) whether processor design of the Intel architectures has advanced since 2005 in a way that allows processor-hardware-based detection of running these instructions in user mode? If so, cite specific sections and terms in your linked readings that indicate how you arrive at this conclusion.

**Q9:** On mcgonagall, run `ps -Alf | less` or `ps -Alf | grep STRING`, where STRING is a string that you are looking for in the output from `ps`, and find all of the running VirtualBox processes from those enumerated in Section 10.2 of the VirtualBox reference [https://www.virtualbox.org/manual/ch10.html#technical-components](https://www.virtualbox.org/manual/ch10.html#technical-components) or here [http://www.informit.com/articles/article.aspx?p=1627061](http://www.informit.com/articles/article.aspx?p=1627061). Run `man ps` if you need documentation on running the Unix `ps` command and interpreting its output. Copy the lines from running `ps` that relate to VirtualBox and paste them in your answer below. Then, in a line following each, summarize what that process does according to the documentation in the two links in this question.

**Q10:** Section 8.3 of our textbook summarizes the state of virtualizing memory access in VMs in 2005, especially per-VM page table mapping. From the linked readings in this assignment, summarize one hardware assist that has been added since the textbook was written in 2005 to (Intel, AMD, or other) processor and/or memory management unit (MMU) design for page table mapping to physical memory in support of hardware virtualization. Cite your reference source(s) and section or slide numbers used to answer this question.

**Q11:** Section 8.4 of our textbook summarizes the state of virtualizing I/O access in VMs in 2005. From the linked readings in this assignment, summarize one hardware assist that has been added since the textbook was written in 2005 to (Intel, AMD, or other) processor and/or I/O controller-DMA design for mapping I/O accesses to physical I/O in support of hardware virtualization. Cite your reference source(s) and section or slide numbers used to answer this question.

**F. Assignment Grading**

I plan to curve grading on this assignment as described next for assignments turned in on time via D2L. I will deduct the usual 10% per day late penalty **after applying the curve.**

The curve works as follows. I will limit the lowest project grade to 75% as long as all questions are answered seriously, i.e., with an effort. Please answer all questions Q1 through Q11; each is worth 9% of the project, with the remaining 1% thrown in for turning it in. If the lowest grade is > 75%, it will stay as it is. I will scale the highest grade up to 100% unless it is already at 100%. Then I will take grades between the lowest and highest and re-scale them according to these limits. For example, if raw grade A is 65%, raw grade B is 95%, and raw grade C is 75% (1/3 of the way from the lowest to the highest), then curved grade A will be 75%, curved grade B will be 100%, and curved grade C will be 83.33% (again, 1/3 of the way from the the lowest and highest).

No one can lose points in this scheme. I am doing it because I am looking for precise, accurate answers. Some questions may have more than one possible precise answer. That is OK, but I will not accept merely plausible answers or guesses. Those earn 0%. Some questions may allow for partial credit.

Late charges apply at 10% per day to the curved grade. Please turn this in via D2L by the end of December 7.

---

2 DMA is direct memory access as examined in detail in the undergrad CSC343 course.