University POLITEHNICA of Bucharest

Automatic Control and Computers Faculty,
Computer Science and Engineering Department

BACHELOR THESIS

vmchecker - enhancement and scalability

Scientific Adviser:
As. Dr. Ing. Răzvan Deaconescu

Author:
Valentin Gosu

Bucharest, 2012

Abstract

The testing and grading of students' programming assignments is a process that requires a lot of time, but also a similar testing environment and an objective individual to do the testing. Vmchecker is a system that allows the automated testing and grading of programming assignments with little human involvement.

This project is an attempt to improve the system, by expanding the types of testing environments and making the process scalable by distributing tasks between several workstations.

Contents

List of Figures

Notations and Abbreviations

procfs – proc filesystem

Chapter 1

Introduction

TODO:
Write Intro chapter

1.1 Brief history of vmchecker

1.2 The architecture of vmchecker

1.3 Analysis of the shortcomings of the former architecture

1.4 Improving the existing model

Chapter 2

Extending the types of virtualization

TODO:
Write Intro chapter

2.1 LXC

2.2 KVM

2.3 VMware

2.4 vmchecker's executor

Chapter 3

Redesigning the communication model

TODO:
Write Intro chapter

3.1 Overview

3.2 Authentication and security

3.3 Daemon's implementation

Chapter 4

Load balancing and self-management

TODO:
Write Intro chapter

4.1 Estimating the evaluation time

4.2 Choosing the best machine

Chapter 5

Testing and evaluation

TODO:
Write Intro chapter

Chapter 6

Conclusion

TODO:
Write Intro chapter

Chapter 7

Further development

TODO:
Write Intro chapter

Appendix A

Project Build System Makefiles

A.1 Makefile.test

```
# Makefile containing targets specific to testing

TEST_CASE_SPEC_FILE=full_test_spec.odt
API_COVERAGE_FILE=api_coverage.csv
REQUIREMENTS_COVERAGE_FILE=requirements_coverage.csv
TEST_REPORT_FILE=test_report.odt


# Test Case Specification targets

.PHONY: full_spec
full_spec: $(TEST_CASE_SPEC_FILE)
        @echo
        @echo "Generated␣full␣Test␣Case␣Specification␣into␣\"$^\""
        @echo "Please␣remove␣manually␣the␣generated␣file."

.PHONY: $(TEST_CASE_SPEC_FILE)
$(TEST_CASE_SPEC_FILE):
        $(TEST_ROOT)/common/tools/generate_all_spec.py --format=odt
            -o $@ $(TEST_ROOT)/functional-tests $(TEST_ROOT)/
            performance-tests $(TEST_ROOT)/robustness-tests
#

# ...
```

Listing A.1: Testing Targets Makefile (Makefile.test)

Bibliography

[1] Argparse tutorial - python. http://docs.python.org/dev/howto/argparse.html#id1. [Online; accessed 13-06-2012].

[2] Creating a read/write proc entry. http://tuxthink.blogspot.ro/2011/02/creating-readwrite-proc-entry.html. [Online; accessed 22-05-2012].

[3] Creating and using proc files. http://www.rt-embedded.com/blog/archives/creating-and-using-proc-files/. [Online; accessed 22-05-2012].

[4] How to compile linux kernel. http://elf.cs.pub.ro/so2/wiki/laboratoare/lab_compilare. [Online; accessed 20-03-2012].

[5] How to use stacktrace. http://stackoverflow.com/questions/5863122/how-to-include-c-backtrace-in-a-kernel-module-code. [Online; accessed 29-04-2012].

[6] The kernel newbie corner: Kernel debugging with proc "sequence" files–part 1. https://www.linux.com/learn/linux-career-center/37985-the-kernel-newbie-corner-kernel-debugging-using-proc-qsequenceq-files-part-1. [Online; accessed 30-05-2012].

[7] The kernel newbie corner: Kernel debugging with proc "sequence" files–part 2. https://www.linux.com/learn/linux-career-center/39972-kernel-debugging-with-proc-qsequenceq-files-part-2-of-3. [Online; accessed 30-05-2012].

[8] The kernel newbie corner: Kernel debugging with proc "sequence" files–part 3. https://www.linux.com/learn/linux-career-center/44184-the-kernel-newbie-corner-kernel-debugging-with-proc-qsequenceq-files-part-3. [Online; accessed 30-05-2012].

[9] Kobjecs, ksets and subsystem. http://www.makelinux.net/ldd3/chp-14-sect-1. [Online; accessed 28-02-2012].

[10] Sysfs and kobjects. http://www.win.tue.nl/~aeb/linux/lk/lk-13.html. [Online; accessed 27-02-2012].

[11] The zen of kobjects. http://lwn.net/Articles/51437/. [Online; accessed 26-02-2012].

[12] Greg Kroah-Hartman. kobjects and krefs. http://www.kroah.com/linux/talks/ols_2004_kref_paper/Reprint-Kroah-Hartman-OLS2004.pdf. [Online; accessed 27-02-2012].

[13] Paul E. McKenney. Overview of linux-kernel reference counting. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2007/n2167.pdf. [Online; accessed 28-02-2012].

[14] Paul E. McKenney. What is rcu? part2: Usage. http://lwn.net/Articles/263130/. [Online; accessed 20-05-2012].

[15] Corey Minyard. How to use kref structure. http://marcbug.scc-dc.com/svn/repository/trunk/linuxkernel/linux-2.6.16-mcemu/Documentation/kref.txt. [Online; accessed 28-02-2012].

[16] Jonathan Walpole Paul E. McKenney. What is rcu, fundamentally? http://lwn.net/Articles/262464/. [Online; accessed 20-05-2012].

[17] Gilles Muller Suman Saha, Julia Lawall. Finding resource-release omission faults in linux. http://pagesperso-systeme.lip6.fr/Suman.Saha/src/plos11.pdf. [Online; accessed 27-02-2012].