

Hardware Fault Injection with UMLinux

K. Buchacker, M. Dal Cin, H.-J. Höxer, V. Sieh, O. Tschäche, M. Waitz
Institut für Informatik 3
Friedrich Alexander Universität Erlangen-Nürnberg
Germany

1. UMLinux Overview

The UMLinux [3] environment provides virtual machines on top of which the Linux operating system and off-the-shelf software is installed. Using UMLinux, you can set up systems consisting of several networked virtual machines.

A userfriendly GUI guides you through the process of configuring the virtual hardware. Once the hardware is set up, you can proceed to boot the Linux operating system and install off-the-shelf software just as you would on a real machine. Thus popular Linux distributions like RedHat or SuSE and software like the Oracle database system run on top of the virtual machines out of the box. The GUI allows you to control virtual machines interactively, just as though you were sitting in front of them. You have access to (virtual) keyboard, mouse, and monitor as well as the system's drives and on/off buttons. A UMLinux virtual machine is fully network capable and you can connect it to other virtual machines as well as to real machines. The GUI will guide you through the process of setting up the network connections. To analyse a system's behaviour in the presence of faults and to set up worst-case scenarios, you can use the GUI to inject faults in the hardware of a virtual machine.

2. Experiment Automization

UMLinux also supports automization for benchmarking experiments or large scale testing. This automization is script based and once set up, no further user interaction is needed. We have chosen VHDL as language for scripts, as VHDL is standardised, well understood and the semantics of VHDL scripts are unambiguous. Automization is supported in all phases of benchmarking and testing, including configuring the virtual hardware, installing the necessary software, exercising a workload, injecting faults and taking measures. The UMLinux environment [3] is therefore ideal to analyse the behaviour of Linux applications and the Linux kernel itself in the presence of hardware faults.

We are using the UMLinux system to develop dependability benchmarks for the DBench Project [2].

3. Example Scenarios

We have set up a number of experiments using UMLinux. This ranges from very simple tests to complex client-server configurations involving several machines. The automatic installation of the well-known SuSE distribution onto a single machine using the graphical installation wizard provided on the SuSE-CD is a simple example which we used later in other experiments to set up similar machines.

A more complex scenario is a load-balanced webserver-array accessed by a number of clients. The load-balancer distributes the client requests evenly among the real servers and is able to remove a crashed or unresponsive server from the array. The load-balancer itself is duplicated, so that in case one system fails, the backup can take over.

In our largest experiment so far we have modelled an online transaction system consisting of an Oracle database, an application server and several client machines. To analyse the reliability of the Oracle database we injected hard disk and network faults into the virtual machine running the database while executing the TPC-C performance benchmark as workload. The experiments were fully automated using the UMLinux experiment controller and VHDL scripts. One of the results was, that in over 80 percent of the hard disk faults, there was no visible effect on the operation of the database. In over 10 percent, the database failed completely, in a few cases database errors were logged. In extremely few cases the database fault-tolerance mechanisms detected and correctly recovered from the error [1].

References

- [1] K. Buchacker, M. Dal Cin, H. Höxer, R. Karch, V. Sieh, and O. Tschäche. Reproducible dependability benchmarking experiments based on unambiguous benchmark setup descriptions. In *Proceedings of the International Conference on Dependable Systems and Networks*, 2003.
- [2] DBench - Dependability Benchmarking (Project IST-2000-25425). URL: <http://www.laas.fr/DBench/>, 2001.
- [3] UMLinux Team. UMLinux. URL: <http://umlunix.de/>, 2002.