Testing Device Drivers against Hardware Failures in Real Environments

Satoru Takekoshi
University of Tsukuba

Takahiro Shinagawa
The University of Tokyo

Kazuhiko Kato
University of Tsukuba

The reliability of computer systems has become more important as the dependency on computer systems for our modern lives increases. Crashes of computer systems can results in massive financial or life loss, even with tens of minutes of downtime. Among others, hardware failures are a major cause of system crashes [4]. Unfortunately, with the wide spread of low-cost commodity hardware products, hardware failures has become inevitable. Therefore, software of computer systems should tolerate hardware failures as much as possible to improve reliability.

Device drivers are known to be the weakest components of operating systems. A study on operating system errors [10] reported that Linux drivers had an error rate up to 7 times higher than the rest of the kernel. On Windows Vista, millions of crashes were caused by drivers [11]. Moreover, many device drivers are vulnerable to hardware failures. A study by Microsoft [9] showed that 9% of unplanned reboots of Windows servers were due to driver or hardware failures and the majority of failures were transient. This study also showed that fault tolerant systems could decrease storage and network adapter failure rates from 8% to 3%, and claimed that drivers could mask the effects of device failure by its design. These evidences imply that (1) poorly implemented drivers are used widely as a part of products, and (2) drivers can survive some of hardware failures if properly implemented.

A previous work used source code analysis to find inappropriate handling of hardware failures [4]. Unfortunately, this approach cannot test close-source drivers and has the possibility of false detection. Several works used symbolic execution to explore the entire code for testing [5, 6, 7]. However, they require virtual environments specially built for them and has the possibility of missing side-effect behavior that only occurs in real environments.

This paper presents FaultVisor, a bare-metal hypervisor for testing device drivers against hardware failures via fault injection. FaultVisor does not virtualize devices and allows pass-through access to real hardware, while it slightly modifies access to the hardware to simulate hardware failures. In cooperation with controller software running in user mode, target devices and fault patterns can be configured in runtime easily. In addition, testing can be repeatedly performed automatically, and the test cases are reproducible. FaultVisor is transparent from the guest OS and kernels, and drivers do not need to be modified. It is also easily inserted into existing systems to test device drivers in real environments.

We implemented FaultVisor based on BitVisor [3], a versatile platform for mediating device access used for various purposes [1, 2]. In current evaluation, FaultVisor and manual analysis based on the test results identified 41 problems. 30 of the identified problems caused critical system failures such as crashes or hangs.

References

[1] Yushi Omote et al. “Improving Agility and Elasticity in Bare-metal Clouds.” In Proceedings of the 20th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS ’15), 145-159, March 2015.

[2] Yohei Matsuhashi et al. Transparent VPN Failure Recovery with Virtualization. Future Generation Computer Systems, Elsevier, Vol. 28, No. 1, pp. 78-84, Jan 2012.

[3] Takahiro Shinagawa et al. BitVisor: A Thin Hypervisor for Enforcing I/O Device Security. In Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2009), pp. 121-130, Mar 2009.

[4] Asim Kadav et al. ”Tolerating hardware device failures in software.” Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles. 2009.

[5] Kuznetsov et al. ”Testing Closed-Source Binary Device Drivers with DDT.” USENIX Annual Technical Conference. 2010.

[6] Vitaly Chipounov et al. ”S2E: A Platform for In Vivo Multi-Path Analysis of Software Systems”, Proceedings of the 16th Intl. Conference on Architectural Support for Programming Languages and Operating Systems. 2011.

[7] Matthew J. Renzelmann et al. ”SymDrive: testing drivers without devices.” Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation. 2012.

[8] Man-Lap Li et al. ”Understanding the propagation of hard errors to software and implications for resilient system design.” ACM SIGARCH Computer Architecture News. Vol. 36. No. 1. ACM, 2008.

[9] S. Arthur. ”Fault resilient drivers for Longhorn server”, WinHec 2004 Presentation DW04012, 2004

[10] Andy Chou et al. ”An empirical study of operating systems errors.” Vol. 35. No. 5. ACM, 2001.

[11] Microsoft. Microsoft internal memo, disclosed as public evidence in court case 2:07-cv-00475-MJP, Kelley v. Microsoft Corporation Filing 131. `https://docs.justia.com/cases/federal/district-courts/washington/wawdce/2:2007cv00475/142597/131` 2008.