LLFI: A High-Level Fault Injection Framework for Evaluating Software Error Resilience Techniques

Karthik Pattabiraman
Electrical and Computer Engineering, University of British Columbia (UBC)
{karthikp}ece.ubc.edu

Abstract

Fault-injection has traditionally been performed at lower levels of the system stack, such as in the microarchitectural or circuit levels. Such injection techniques are however difficult to use for evaluating the coverage of software error resilience mechanisms. In this talk, I present *LLFI*, a high-level fault injection mechanism for evaluating the coverage of software resilience mechanisms. LLFI is highly flexible, easy to use and allows mapping between the application's code and fault propagation. It is also accurate with regard to assembly-code level fault injection for Silent Data Corruption (SDC) causing errors.

Keywords Reliability, Compiler, Modeling

Fault-injection is the de-facto method for evaluating the error resilience of a system. Fault-injection involves the perturbation of the system in a controlled manner so as to study the effect of the introduced fault, and to test the resilience of the system to the fault. Traditionally, fault injection has been performed at the micro-architectural or circuit level, as this is where (hardware) errors can be easily modelled. Further, error resilience mechanisms have been traditionally implemented in the hardware, and hence injecting faults at the microarchitectural or circuit levels made sense.

However, recently there has been a trend towards incorporating software error resilience mechanisms at the application level [1–3, 5–8]. There are two reasons for this trend. First, many errors do not propagate up the system stack to the application, and hence it is much more cost-effective to protect the application from the few errors that do propagate. Second, it is possible to take into account the properties and requirements of applications when designing error resilience mechanisms for them. For example, a video processing application is much more error resilient than an application that processes financial transactions, and hence requires lower levels of protection. Therefore, application-level resilience techniques that consider the application's properties can be much more efficient than generic hardware-level techniques.

The main challenge with application-level error resilience mechanisms is that there is a lack of robust fault-injection frameworks for evaluating their coverage. This is because unlike generic mechanisms, application-level mechanisms are intricately tied to the properties of the application they protect, and hence their coverage needs to be evaluated in the context of the application's code. Further, fault-injection approaches at the microarchitectural or circuit levels are often too slow to evaluate the end-to-end coverage of applications as they are implemented in microarchitectural or circuit simulators. Finally, performing fault injection at lower levels of the system stack often provides very little insight to the application developer on why the resilience mechanism does not provide the desired coverage, and to debug the resilience mechanism itself.

In this talk, I will present *LLFI*, a high-level fault-injection framework to evaluate the coverage of application-level error-resilience techniques. LLFI[1] integrates with the widely used LLVM compiler [4], and provides easy mapping between the application's LLVM intermediate representation (IR) and its error resilience properties. Further, it allows programmers to control the injection of faults into selected program segments and/or data regions, and provides a programmable interface to choose different fault-models for the program. Finally, LLFI traces the propagation of faults in the program (after they have been injected) to provide insights into how and why faults propagate in the program, and whether the faults are detected by the software resilience mechanisms. LLFI is easy to install and use, and has been tested on a wide variety of platforms and architectures.

We have compared the accuracy of *LLFI* with other application-level fault injectors that operate at the assembly code levels of programs. We find that LLFI is accurate in evaluating the effects of Silent Data Corruption (SDC) causing errors compared to assembly level injectors, but not for crash-causing errors [9]. This result holds across different categories of instructions in the program. I will conclude by outlining some of the challenges and opportunities in the area of high-level fault injection frameworks.

References

[1] J. Cong and K. Gururaj. Assuring application-level correctness against soft errors. In *ICCAD*, 2011.
[2] M. de Kruijf, S. Nomura, and K. Sankaralingam. Relax: An architectural framework for software recovery of hardware faults. In *ISCA.*, 2010.
[3] S. Feng, S. Gupta, A. Ansari, and S. Mahlke. Shoestring: probabilistic soft error reliability on the cheap. In *ASPLOS*, 2010.
[4] C. Lattner and V. Adve. LLVM: A compilation framework for lifelong program analysis & transformation. In *CGO.*, 2004.
[5] K. Lee, A. Shrivastava, I. Issenin, N. Dutt, and N. Venkatasubramanian. Mitigating soft error failures for multimedia applications by selective data protection. CASES '06, pages 411–420, 2006.
[6] Q. Lu, K. Pattabiraman, M. S. Gupta, and J. A. Rivers. Sdctune: A model for predicting the sdc proneness of an application for configurable protection. In *CASES*, 2014.
[7] M. Shafique, S. Rehman, P. V. Aceituno, and J. Henkel. Exploiting program-level masking and error propagation for constrained reliability optimization. In *DAC*, 2013.
[8] A. Thomas and K. Pattabiraman. Error detector placement for soft computation. In *DSN*, 2013.
[9] J. Wei, A. Thomas, G. Li, and K. Pattabiraman. Quantifying the accuracy of high-level fault injection techniques for hardware faults. In *DSN*, 2014.

ESWEEK'14, October 12 - 17 2014, New Delhi, India.

http://dx.doi.org/10.1145/2656106.2656127

[1] Available at *https://github.com/DependableSystemsLab/LLFI*