Fault Injection with FAIL*

DSN '18 Hands-On Tutorial

Horst Schirmeier and Olaf Spinczyk
Department of Computer Science 12
Technische Universität Dortmund, Germany
e-mail: {horst.schirmeier, olaf.spinczyk}@tu-dortmund.de

Fault injection (FI) has been a standard technique for test, measurement, and comparison of fault-tolerance implementations for decades. By using, for example, a virtual machine emulating faulty hardware, the developer of a software-implemented fault tolerance method can test the implementation, and measure its effectiveness in an adverse environment. FI can also be used to emulate other kinds of faults, such as communication problems, or even programming errors (bugs).

In this hands-on tutorial, we will focus on software-implemented hardware fault tolerance (SIHFT), and introduce the participants to simulation-based hardware FI for analysis, test and measurement. The open-source FI framework FAIL* will be used to analyze a small example program. Working in groups or autonomously, the participants will – choosing from given suggestions, or inventing their own SIHFT technique – improve the benchmark on the source-code level to make it more resilient against hardware faults. In the grand finale, all solutions will compete for a small prize and, certainly, fame!

I. Motivation and Goals

FI is a frequently used technique in the DSN community to evaluate the resilience of hardware or software and the effects of new fault tolerance mechanisms. As FI itself is not their main focus, many researchers use simplistic old or home grown tools, which reduces the quality of the results or consumes more resources than necessary. FAIL* [1] is an open source tool, which could help to avoid "reinventing the wheel" in many cases. It has been used by many research groups in several different use cases successfully. It has been developed during the last five years, is continuously maintained, and runs on up-to-date Linux platforms. Its ability to execute FI campaigns on compute clusters in parallel makes it an ideal tool for larger experiments or for experiments with full fault-space coverage.

The main goal is to give the participants a first, hands-on experience with the FAIL* FI tooling. This shall make sure they have a steep learning curve when using FAIL* for their own research, for instance, to evaluate novel SIHFT mechanisms. Each participating team will iteratively use FAIL*:

- An *analysis* of the target program reveals particularly "weak" spots, e.g. specific data structures (see Fig. 1) or program modules that should be protected with SIHFT at the highest priority.
- The team hardens the program by, e.g., adding redundancy to weak spots, or improving the program algorithmically.

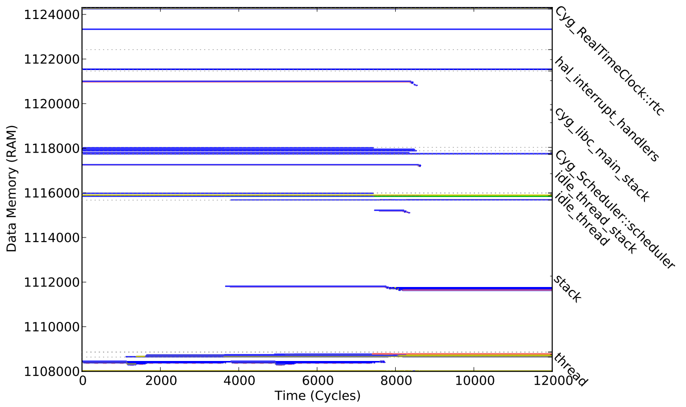

Fig. 1. Fault-space plot of an FI campaign on an embedded OS kernel, produced by FAIL* [2].

- *Test, measurement* and *comparison* with earlier versions of the program determines whether the SIHFT implementation works correctly, and whether it actually improves fault tolerance in this setting [3].

II. Organization

We are planning a compact (3 hours) introductory tutorial:

- It briefly introduces the concepts and pros/cons of SIHFT,
- familiarizes participants with FI concepts, techniques, tools, and pitfalls, and
- is fun, because simple fault-tolerance mechanisms are evaluated "hands-on" with FAIL* in an interactive manner.

The target audience are developers and researchers from industry and academia (of course, including students generally interested in the presented topics). The tutorial is intended for beginners in the field of SIHFT and FI. However, some experience with C/C++ and working on the Linux commandline is expected.

References

[1] H. Schirmeier, M. Hoffmann, C. Dietrich, M. Lenz, D. Lohmann, and O. Spinczyk, "FAIL*: An open and versatile fault-injection framework for the assessment of software-implemented hardware fault tolerance," in *11th Europ. Depend. Comp. Conf. (EDCC '15)*. IEEE, Sep. 2015.

[2] C. Borchert, H. Schirmeier, and O. Spinczyk, "Generative software-based memory error detection and correction for operating system data structures," in *43rd IEEE/IFIP Int. Conf. on Dep. Sys. & Netw. (DSN '13)*. IEEE, Jun. 2013.

[3] H. Schirmeier, C. Borchert, and O. Spinczyk, "Avoiding pitfalls in fault-injection based comparison of program susceptibility to soft errors," in *45th IEEE/IFIP Int. Conf. on Dep. Sys. & Netw. (DSN '15)*. IEEE, 2015.