# Chapter 11
# Analyzing the Robustness of HPC Applications Using a Fine-Grained Soft Error Fault Injection Tool

**Qiang Guan**
*Los Alamos National Laboratory, USA*

**Song Fu**
*University of North Texas, USA*

**Nathan DeBardeleben**
*Los Alamos National Lab, USA*

**Claude H. Davis IV**
*Clemson University, USA*

**Sean Blanchard**
*Los Alamos National Lab, USA*

**William M. Jones**
*Coastal Carolina University, USA*

## ABSTRACT

*As the high performance computing (HPC) community continues to push towards exascale computing, HPC applications of today are only affected by soft errors to a small degree but we expect that this will become a more serious issue as HPC systems grow. We propose F-SEFI, a Fine-grained Soft Error Fault Injector, as a tool for profiling software robustness against soft errors. We utilize soft error injection to mimic the impact of errors on logic circuit behavior. Leveraging the open source virtual machine hypervisor QEMU, F-SEFI enables users to modify emulated machine instructions to introduce soft errors. F-SEFI can control what application, which sub-function, when and how to inject soft errors with different granularities, without interference to other applications that share the same environment. We demonstrate use cases of F-SEFI on several benchmark applications with different characteristics to show how data corruption can propagate to incorrect results. The findings from the fault injection campaign can be used for designing robust software and power-efficient hardware.*

DOI: 10.4018/978-1-5225-0287-6.ch011

## INTRODUCTION

*If you cannot measure it, you cannot improve it. – Lord Kelvin*

Exascale supercomputers are likely to encounter failures at higher rates than current high performance computing systems. Next generation machines are expected to consist of a much larger component count than current petascale machines. In addition to the increase in components, it is expected that each individual component will be built on smaller feature sizes, which may prove to be more vulnerable than current parts. This vulnerability may be aggravated by near-threshold voltage designs meant to dramatically decrease power consumption in the data center (Kaul et al., 2012). Due to high error rates it is estimated that exascale systems may waste as much as 60% (DeBardeleben et al., 2009) of their computation cycles and power due to the overhead of reliability assurance. These high error rates pose a serious threat to the prospect of exascale systems.

Soft errors fall into three categories (Snir et al., 2014): DCE (Detected and Corrected Error), DUE (Detected but Uncorrectable Error) and SE (Silent Error). Most DRAM in supercomputers is protected by Chipkill (Jian, Duwe, Sartori, Sridharan, & Kumar, 2013; Sridharan, Stearley, DeBardeleben, Blanchard, & Gurumurthi, 2013; Walker & Betz, 2013), which makes DUE events rare and SE events even more rare. SRAM in cache layers, however, is generally protected by SECDED (Single Error Correction and Double Errors Detection) or parity and is therefore more vulnerable to SE events. In addition, logic circuits have varying levels of internal protection and we expect these error rates to be on the rise as well.

Silent errors pose a serious issue when they lead to Silent Data Corruption (SDC) in user applications. If undetected by the application, a single SDC can corrupt data causing applications to output incorrect results malfunction or hang. Unfortunately, detecting and correcting SDC events at the application layer is challenging. It requires expert knowledge of the algorithm involved to determine where an application might be most vulnerable and how it will behave if an SDC should occur. Even with such knowledge it is difficult to test any mitigation techniques an application author might attempt since SDC events occur rarely and in most cases randomly.

In order to facilitate the testing of application resilience methods, detection and correction mechanisms require the study of hardware faults that may affect the execution of any instruction of the target application. However, a brute-force way that explores billions of instructions of an application or benchmark may result in trillions of vulnerable sites and requires a prohibitively large number of experiments to cover all possible cases. In order to perform the fault injection efficiently and effectively, at least three aspects need to be clarified.

- **Where to Inject:** The fault injector should allow users to define the location of the errors to be injected, e.g., application, function, line-of-code, even variable and register.
- **When to Inject:** The fault injector should grant the ability to users to fully control the injection at runtime to investigate the susceptibility of application under different stages.
- **How to Inject:** The fault injector should provide different fault models to users with different interests.

Therefore, we present a fine-grained soft error fault injector named F-SEFI, which is built on the previous work (DeBardeleben, Blanchard, Guan, Zhang, & Fu, 2011; Guan, Fu, DeBardeleben, & Blanchard, 2014; Guan, DeBardeleben, Blanchard, & Fu, 2014). F-SEFI allows for the targeted injection