Ex-Post Hardening Operating Systems to Survive Main-Memory Errors

Christoph Borchert, TU Dortmund

Abstract

Errors in main memory are one of the primary hardware problems for failures of today's computer systems. Recent studies on current DRAM technology (DDR2 and DDR3) confirm an approximate fault rate of 0.066 FIT/Mbit. For a computing cluster with terabytes of main memory, this fault rate translates to one memory failure every few hours. The ever increasing demand for higher memory capacities worsens this reliability problem.

In general, the operating-system (OS) *kernel* is the most important piece of software with regard to dependability, as all other software components depend on the OS. Surprisingly, in spite of their impact on total system resiliency and – compared to the rest of the system – their very small memory footprint, state-of-the-art OS kernels are not equipped with software-based protection against memory errors. This fact may be contributed to the tedious task of manually implementing software-based error-detection mechanisms (EDMs) and error-recovery mechanisms (ERMs) in the OS kernel. An unmanageable amount of source code needs to be rewritten and verified. In particular, special care has to be taken that the inherent kernel synchronization is not violated.

However, the conceptual *algorithms* used for memory-error handling remain similar across the different source-code locations in the kernel code, for example, inserting a checksum to protect important data structures. *Only the adaptation* of such algorithms to the particular data structures depends on the respective source-code locations. Modern programming paradigms, such as *Aspect-Oriented Programming* with AspectC++, offer means to express crosscutting concerns in a single modular source-code file. Thus, an EDM/ERM can be implemented (and verified) *once*, and can then be applied *often* to several OS code locations. The adaptation of the EDM/ERM to the specific OS code locations is carried out by the AspectC++ compiler using its feature to *introspect* the source code at compile time. Thereby, EDM/ERM implementations can even be *reused* for multiple operating systems.

In this talk, I am going to discuss our experiences on ex-post hardening the embedded operating system eCos. We designed and implemented several aspect-oriented EDMs/ERMs that can be selectively applied to the kernel data structures. I will discuss the trade offs between runtime overhead and reliability gains, substantiated by extensive fault-injection campaigns. Finally, I will conclude my talk with an outlook on hardening the Fiasco/L4 microkernel with the very same methodology.