Revisiting OS Multi-Calls in the Light of Meltdown and KPTI

Horst Schirmeier, TU Dortmund,
Embedded System Software Group
horst.schirmeier@tu-dortmund.de

Context switches between a non-privileged application process and the OS kernel running in the CPU's supervisor mode have traditionally been costly, despite manufacturer efforts to provide specially optimized instructions for this transition. Kernel page-table isolation (KPTI), the now widely deployed mitigation for the Meltdown vulnerability in Intel (and other) microprocessors, dramatically increased this cost: directly, by switching the active page table at each context transition (i.e., two page-table switches per system call), and indirectly by the subsequent, longer-lasting performance penalty from the TLB flushes triggered by these address-space changes. In consequence, particularly system-call–heavy applications have been reported to be slowed down by up to 30 percent on KPTI-protected systems.

In this talk I will revisit an old idea from the OS community known as "system-call clustering", "system-call batching", or "multi-calls": the bundling of several successive but potentially not directly related system calls into a combined call, which only incurs the kernel-transition costs of a single system call. I will discuss earlier approaches, including compiler-based techniques that make multi-calls transparent to the application developer. I will outline how these approaches could be mapped to the current Linux ecosystem in the light of Meltdown and Spectre, be extended to cover the most KPTI-affected application patterns, and effectively could eliminate the slowdown. Additionally, I will present first results of a feasibility study highlighting system-call patterns and potential performance gains in a real-world application.