Advanced Linux Kernel Programming
FlexSC- Flexible System Call Scheduling with Exception-Less System Calls
paper summary

Kolloju Kashi Vishwanath

In this paper, a new mechanism on how to implement Exception less System Calls has been proposed. For many years, we have been using the traditional way of implementing system calls to enter into kernel space from User space that are Synchronous. But with this synchronous system calls it affects the performance of the system particularly on heavy workloads. These performance costs include Mode Switching, System call footprint, Impact of System Call on user mode IPC (instructions per cycle) and Mode Switching impact on kernel IPC. To counter the performance effects of Synchronous System Calls, Exception less system call - **FlexSC** technique is proposed and is implemented by delaying the execution of system calls and executing the batched system calls reduces the frequency of switching between user mode and kernel mode. FlexSC also exploits multi-core systems by allowing a system call scheduled on a core that is different from the core it is called. To perform system call batching, memory pages that are shared by user space and kernel space called as syscall page is implemented. These pages contain the system call requests, arguments, status and result. Moreover, to achieve syscall page method the kernel must support delayed execution of system calls. For executing this a new kernel threading mechanism – **syscall threads** are used that only execute in kernel space and these threads get requests from syscall pages and execute them instead of user mode thread. The paper described the design and implementation of **FlexSC-Threads**, a threading package that transforms synchronous system calls into exception-less ones transparently to applications mainly server type applications such as Apache and MySQL. FlexSC threads provides multicore support by creating a single kernel visible thread per core available to the process. A syscall thread scheduler that manages which system calls execute on which cores as it helps in the flexibility and locality of user and kernel space. For evaluating the performance of FlexSC two server applications, Apache and MySQL are tested on Linux kernel (minimum performance results were collected using Linux default native POSIX threading library-NPTL). With this implementation, the paper presented how FlexSC improves throughput of Apache by up to 116% and MySQL by up to 40% without ant modifications to the applications. While having to increase processor efficiency the drawback of implementing asynchronous system calls is that kernel cannot process the system call request immediately. In addition, use of multi threads pose a security threat to the system as many threads are used to switch from user space to kernel space and vice-versa. Finally, we cannot determine when the system call is going to be executed unlike the synchronous system calls.