Zhenyu Yan

The benefits and costs of writing a POSIX kernel in a high-level language

From

Cody Cutler, M. Frans Kaashoek, Robert T. Morris

In this paper, The authors of this paper, Cutler, Kaashoek, and Morris of MIT, created Biscuit, a POSIX kernel written almost entirely in a high level language. They shows the evaluation uses a high-level language (HLL) with garbage collection to implement a single POSIX-style kernel. The goal is to explore the feasibility of using HLL instead of C to implement such a kernel by examining performance costs, implementing challenges, programmability, and security advantages.

First, they talks about motivations and methodologies, One of the main motivations of the author to write a kernel in HLL is automatic memory management, making programming easier and reducing errors. Another motivation for the authors to try to use HLL is the benefits of integrating garbage collectors for concurrency. Then they talks about the overview of biscuit, such as boot and go runtime, processes and kernel goroutines, interrupts, multi-core and synchronization, virtual memory, file system, network stack and limitations. Thirdly, they talks about garbage collection and how to avoiding head exhaustion. After that, they talks about how biscuit kernel implementations. At the end, they shows the evaluation about the performance, discussion and future work. Languages that don't have the same compiler as Go, or languages that are designed to be removed from the underlying machine may not perform well. On the other hand, languages like Rust that avoid garbage collection may provide higher performance and security, although at the expense of programmability of thread code.

In conclusion, Go's high-level language features are useful in the context of the kernel. Checking for historical Linux kernel errors caused by C suggests that types and memory-safe languages like Go may avoid real-world errors or handle them cleaner than C. The performance costs of Biscuit's use of Go's HLL functionality in a set of kernel-intensive benchmarks. Garbage collection and security checks consume less than 15% of the CPU time. It also compares the performance of equivalent kernel code paths written in C and Go and finds that the speed of the C version is increased by about 15%.

I think this article is very useful because this paper was published only 2 months ago. Also, I believe efficiency is very important now days, and the high-level language can avoid error and handle them cleaner than C. Since C user is decling in these years, people are switching to other programing language, we should consider to use high level language to implement kernel.

The presenter Michael Mappes and Trey Miller were good. Their slide were detail and include some tables , graphs and pictures. Their arrange their slide left and right and it looks very clean and comfortable. Although, Michael Mappes and Trey Miller did not have enough time to explain more detail and relative work about this paper, overall I learn a lot form them.