Finesse

Tony Mason
University of British Columbia
fsgeek@cs.ubc.ca

Matheus Stolet
University of British Columbia
stolet@cs.ubc.ca

Abstract

In-kernel file systems are challenging to develop. Programmers involved in kernel development have to deal with bugs that cause fatal system crashes. Userspace file systems mitigate this problem by allowing developers to anchor themselves in a forgiving user-space environment. Unfortunately, popular solutions to file systems in user-space, such as FUSE [1], add significant overhead to operations, making them unpalatable to many applications. Finesse solves this problem by adding a client library and a thin software layer that sits on top of the FUSE library. The Finesse layer allows the user to bypass the kernel for selected operations, so that applications communicate directly with the FUSE file system. New opperations can be added to the application library, enabling applications to extend the file system by creating their own set of custom operations. Ultimately, Finesse is a kernel bypass system meant to improve the performance of user-space file systems, without increasing development complexity.

Finesse achieves better performance by modifying FUSE to serve multiple message sources. This is done by implementing a message passing interface that allows applications to make direct calls to the FUSE file system, effectively avoiding the convoluted kernel path taken by FUSE. If an operation is not implemented in the Finesse layer, it falls back to regular FUSE behaviour, allowing the file system to serve operations coming from Finesse and the FUSE kernel driver.

There are two major pieces to the Finesse system: the Finesse application library and the Finesse FUSE extension. The Finesse application library is a user-facing library that allows the explicit invocation of operations implemented in the Finesse library and the implicit invocation of operations through LD_PRELOAD, that loads the Finesse library before other shared libraries such as *libc*. The Finesse FUSE extension lies between the FUSE file system and the FUSE library. It listens to incoming file system operations from the Finesse library and redirects them to the FUSE file system.

Finesse is still in development, and we are working at porting a bigger number of file system operations to use the Finesse library instead of defaulting to regular FUSE behaviour. Preliminary evaluations have showed that the implementation of *unlink* using the Finesse library led to considerable performance advantages. For example, when deleting 4 million preallocated 4KB files over many directories, regular FUSE had a 14.94% performance overhead, while FUSE with Finesse led to a 29.21% performance gain.

Keywords file systems, operating systems, user-space, kernel bypass

References

[1] Nikolaus Rath. [n. d.]. FUSE: Filesystem in Userspace. https://github.com/libfuse/libfuse. ([n. d.]).

Conference'17, July 2017, Washington, DC, USA
2020. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn