File Systems as Processes

Summary:
Submitted by: Sharanya Kamath (16CO140)

Accessing fast storage without kernel involvement has been a central concern for many years. In this paper, it is argued that file systems as processes may be one avenue for progress in this important direction. The study on the challenges of realizing such a file system reveals some promise, but many future hurdles must be overcome to fully reap the benefits that this new idea promises.

The paper implements a prototype of file system process, DashFS, to further examine the issues in realizing the FSP approach. The following design principles are followed:
Trust boundary: DashFS is regarded as a trusted extension of the kernel. User applications are not trusted and must be isolated from other users and the DashFS address space.
Minimal kernel interaction: Kernel involvement is heavyweight and should be avoided. The number of traps into the kernel should not scale with I/O requests.
Concurrency awareness: To provide a fully functional file system, DashFS should be designed with sharing in mind and scale gracefully under load.

DashFS is implemented by extending the user-level NVMe driver provided by SPDK. The communication channel is based upon shared memory, organized as several request buffers, data buffers, and a lockless request queue. During initialization, an application sends a message via UNIX domain socket to DashFS. The kernel (while processing the socket) copies the application's credential information to the DashFS address space from an in-kernel process structure, allocates a shared memory segment for the private communication channel, and returns a key for the channel to the user application. In the block I/O layer (BIO) of our prototype, each thread in Frontend synchronously polls for completion of a request.

This experiments in the paper demonstrate that with an increasing number of user threads, the communication channel scales well in terms of both latency and throughput. Based on the paper's preliminary results, the communication channel between the file system process and application threads can be regarded as efficient enough to support the performance goals of file system process.

Questions:

1. How are the file operations managed in the event where DashFS crashes?
2. Only a few challenges to FSP are discussed. What are the other challenges?