DMFS - a Data Migration File System for NetBSD

William Studenmund

November 29, 1999

Veridian MRJ Technology Solutions
NASA/Ames Research Center

I have recently developed *dmfs*, a Data Migration File System, for NetBSD[1]. This file system is based on the *overlay* file system, which is discussed in a separate paper[2], and provides kernel support for the data migration system being developed by my research group here at NASA/Ames. The file system utilizes an underlying file store to provide the file backing, and coordinates user and system access to the files. It stores its internal meta data in a flat file, which resides on a separate file system.

Our data migration system provides archiving and file migration services. System utilities scan the dmfs file system for recently modified files, and archive them to two separate tape stores. Once a file has been doubly archived, files larger than a specified size will be truncated to that size, potentially freeing up large amounts of the underlying file store. Some sites will choose to retain none of the file (deleting its contents entirely from the file system) while others may choose to retain a portion, for instance a preamble describing the remainder of the file[1]. The dmfs layer coordinates access to the file, retaining user-perceived access and modification times, file size, and restricting access to partially migrated files to the portion actually resident.

When a user process attempts to read from the non-resident portion of a file, it is blocked and the *dmfs* layer sends a request to a system daemon to restore the file. As more of the file becomes resident, the user process is permitted to begin accessing the now-resident portions of the file.

For simplicity, our data migration system divides a file into two portions, a resident portion followed by an optional non-resident portion. Also, a file is in one of three states: fully resident, fully resident and archived, and (partially) non-resident and archived. For a file which is only partially resident, any attempt to write or truncate the file, or to read a non-resident portion, will trigger a file restoration. Truncations and writes are blocked until the file is fully restored so that a restoration which only partially succeed does not leave the file

[1]The specification of any such preamble is left to the applications writing the files as dmfs does not modify the file contents

in an indeterminate state with portions existing only on tape and other portions only in the disk file system.

We chose layered file system technology[3] as it permits us to focus on the data migration functionality, and permits end system administrators to choose the underlying file store technology. We chose the *overlay* layered file system instead of the *null* layer for two reasons: first to permit our layer to better preserve meta data integrity and second to prevent even root processes from accessing migrated files. This is achieved as the underlying file store becomes inaccessible once the *dmfs* layer is mounted.

We are quite pleased with how the layered file system has turned out. Of the 45 vnode operations in NetBSD, 20 (forty-four percent) required no intervention by our file layer - they are passed directly to the underlying file store. Of the twenty five we do intercept, nine (such as vop_create()) are intercepted only to ensure meta data integrity. Most of the functionality was concentrated in five operations: vop_read, vop_write, vop_getattr, vop_setattr, and vop_fcntl. The first four are the core operations for controlling access to migrated files and preserving the user experience. vop_fcntl, a call generated for a certain class of fcntl codes, provides the command channel used by privileged user programs to communicate with the *dmfs* layer.

References

[1] http://www.netbsd.org/

[2] William R. Studenmund. Vnode Layering Changes in NetBSD Version 1.5. Submitted for presentation at USENIX 2000.

[3] John Shelby Heidemann. Stackable Design of File Systems. Ph.D. dissertation, University of California, Los Angeles, 1995.