Vnode Layering Changes in NetBSD Version 1.5

William Studenmund

November 29, 1999

Veridian MRJ Technology Solutions
NASA Ames Research Center

In this paper, I describe changes to the vnode layer in NetBSD[1] between versions 1.4 and 1.5 to assist layered filesystems. The major feature of these changes on the whole is that layered filesystems are now sufficiently robust for production applications[3]. These changes divide readily into three groups: modifications to vnode locking along the lines suggested by Heidemann in [2], changes to the vnode locking protocol, and the introduction of an overlay filesystem. In addition to describing these changes, I will describe the testing we have performed.

When starting work on the data migration layer [3], I evaluated a number of technologies. The stacked file systems of NetBSD looked promising, especially the null layer, nullfs. Unfortunately it did not perform well under load. Multiple concurrent access, either simultaneously accessing the stacked and the underlying layer or multiple accesses of the upper layer (say with a multiprocess make invoked with the **make -j** command) would result in kernel panics due to locking protocol errors.

I made two main changes to the vnode locking implementation. The first was to ensure that nullfs respected the locking protocol: that it obtains all locks from bottom to top and that the VOP_LOOKUP operation foregoes attempting to generate a null layer vnode when looking up the "." path component.

The more dramatic change is to implement the shared vnode lock scheme described by Heidemann[2] as part of his coherence management proposals. With this change, rather than having a vnode (or the associated fs-private storage as done by FreeBSD[4]) provide a lock structure for use by the lock manager, it provides a pointer to a lock structure. For a leaf node (a node on a typical filesystem), it points to the lock structure in that vnode. For a vnode in a stacked filesystem, it points to the lock structure in the underlying layer. In this manner, the entire stack will lock and unlock at the same time. Additionally, a layered filesystem only needs this pointer to perform locking operations - it does not need to call underlying filesystem layers. If a filesystem performs vnode locking other than a call to the lock manager (such as the union filesystem which merges two filesystems, or the NFS filesystem which performs no vnode

locking), it exports a NULL pointer and layered filesystems explicitly call its locking routines.

Two aspects of NetBSD's vnode locking protocol make proper layered filesystem operation challenging. The first is that there was a flaw in the VOP_LOOKUP operation's error case handling. In case of a returned error, the vnode for the parent directory is defined as being locked. The flaw is that looking up the ".." path component requires unlocking and re-locking the parent directory. In case of an error re-locking the parent, there was no way to communicate to the caller that the parent was unlocked. If one of the traversed layers is a union layer (a layer which breaks the ability to share one common vnode lock), then vnode layers stacked above it can get into inconsistent lock states and create deadlock situations. This potential is eliminated by adding a new flag to those passed as part of the lookup operation, PDIRUNLOCK. This flag being set upon exit to the lookup call indicates that the parent directory was unlocked, preventing such inconsistent states. Additionally the layered lookup routine is simplified as this flag is set even when not looking up the ".." path component.

The second difficult aspect is that a number of VOP operations will automatically delete a reference count on a passed-in vnode. This behavior simplifies a number of caller routines, but greatly complicates layered filesystems. It has been worked around by layered filesystems explicitly adding references to a vnode before such a VOP operation down to the underlying layer. This behavior is inefficient, and does not scale well to many layers stacked atop each other. Additionally certain operations which create new vnodes (VOP_CREATE for example) will also delete the reference to the newly created vnode. There is an on-going project which is scheduled to be merged into the NetBSD source for 1.5 to change all such operations to no longer automatically release vnode references. FreeBSD has already merged in many such changes.

The third area of layering change is the introduction of the *overlay* layered filesystem. It is similar to the *null* layer, except that it does not create a new image of the underlying vnodes, it places itself between the underlying filesystem and all future access. This layer is very useful for a certain class of problems where the layered filesystem needs to strictly control access to the underlying files. One such example is the data migration layer we have developed[3] where we need to prevent processes (even root ones) from noticing that we have migrated their files to tape storage. Another application would be the development of new access control or security methods where the layered filesystem could (and definitionaly would need to) totally control access to the files.

As part of this change, most of the null filesystem was moved into a generic set of routines for all layered filesystems which was located in sys/miscfs/genfs. These routines permit the null, umap, and overlay layers to share a large amount of code. Only the individual mount and unmount routines and vfs data structures are not shared.

The tests for these changes were not complicated. As mentioned above, most any simultaneous access would panic a kernel. Thus the simple test regime we used consisted of making multiple *null* mounts of a given leaf filesystem and generating simultaneous access both in all *null* layers and in the underlying

file store. In one filesystem (either the underlying filesystem or one of the *null* layers), I ran a **make -j 10** kernel compile, and in the others I performed multiple hierarchical file accesses, such as **ls -lR**. An unmodified kernel would panic readily (in fact the make -j 10 alone in a null layer would generate a panic), while an updated kernel had no difficulty.

In conclusion, this paper describes changes to the NetBSD vnode system made between versions 1.4 and 1.5 which have greatly improved the stability and robustness of layered filesystems. The method of locking layered vnodes has been improved, the VOP protocol has been modified to not present difficulties to layered filesystems, and a new class of layered filesystems has been introduced with the *overlay* filesystem. On the whole, these changes have opened up new layered filesystem opportunities within NetBSD.

References

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

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

[3] William R. Studenmund. DMFS - a Data Migration Virtual Filesystem Layer for NetBSD. Submitted for presentation at USENIX 2000.

[4] http://www.freebsd.org/