Caching or Not: Rethinking Virtual File System for Non-Volatile Main Memory

Submitted by:
Sharanya Kamath 16CO140

Summary:

This paper presents ByVFS, an optimization of VFS by removing dentry caching to reduce extra overhead for PM file systems. Moreover, it figures out two potential optimization opportunities:

- the concurrency control originally guaranteed in VFS is now shifted to low-level file systems, which is not well supported in existing PM file systems.
- efficiently and completely supporting various system calls, which may require designing highly efficient index structures in persistent memory.

VFS mainly caches three types of metadata, including **superblock**, **dentry**, and **inode**. A dentry metadata (named as dcache in this paper) mainly contains file name and corresponding inode number, while an inode metadata (named as icache in this paper) mainly contains file properties (e.g. inode number and file size).

Path lookup is a critical process involved in many file system operations. Caching both dentry and inode metadata in VFS helps improve path lookup performance for disk-based file systems by avoiding frequently accessing slow disk. However, since NVMs have read latency similar to DRAM, accessing DRAM cache brings performance overhead compared to directly reading NVMs. The paper explores the potential opportunity of directly accessing metadata in PM file systems while bypassing VFS caching.

Following are the design issues of ByVFS, an optimization of Virtual File System for non-volatile memory:

- **Handling dentry cache:** Dentry metadata is not cached and is directly looked in the physical file system. To support getcwd system call, the original VFS's dcache used to maintain a pointer to parent directory so it could iterate dcache to obtain full path. For ByVFS, the authors have provided another iterating lookup approach that traverses the whole directory to serve getcwd.
- **Handling inode cache:** ByVFS caches inode like VFS since it maintains commonly used file properties that are frequently updated. As dcache is removed in ByVFS, so to get the memory location of icache a non-volatile pointer is added to inode structure to point to it. To handle meaningless pointers(pointing to nothing as power loss can remove icache) version numbers are used.
- **Supporting multiple file systems:** A flag is added into the inode of the mounted directory, if it is set ByVFS is used else conventional VFS lookup process is performed.

Questions:

1. The paper focuses on path lookup process for optimization. How does ByVFS work in case of other file system operations optimization?
2. How is concurrency being handled by physical file system?