Abstract

File systems are fundamental to computing, managing data on storage devices, yet they face the persistent challenge of maintaining crash consistency, ensuring data integrity after an unexpected system failure. Operations like file creation or modification require updating multiple, non-adjacent on-disk data structures, and a crash during this process can leave the file system in an inconsistent state, leading to data corruption or loss. While techniques such as journaling improve robustness, verifying crash consistency remains difficult due to system complexity and the vast number of potential failure scenarios. Existing testing methods, including black-box tools like CrashMonkey, suffer from limited coverage and efficiency, often missing subtle corner-case errors. Formal verification with model checking offers exhaustive exploration but has been hampered by state-space explosion and the impracticality of modeling full-stack file systems in languages such as Promela. This thesis introduces a comprehensive methodology and a practical tool, MC^3 (Model Checking for Crash Consistency), for the automated, exhaustive verification of file system crash consistency. Our approach employs a state space search that exhaustively explores file system states while checking correctness properties on-the-fly. To overcome state explosion, we developed two key innovations: an object-based workload generation algorithm that systematically produces all valid file system operations, and a novel directory tree isomorphism detection technique that encodes and eliminates redundant states, drastically reducing the search space. We first applied this methodology to block-based file systems (e.g., FAT, ext2), identified the root causes of consistency violations and the non-atomicity of multi-block metadata updates, and proposed a write-ordering mechanism to prevent critical errors. We then implemented MC^3 to verify file system models written in C/C++. Applying MC^3 to a model of the Flash-Friendly File System (F2FS), a complex log-structured file system (LFS), revealed a previously unknown crash consistency bug where metadata rollback combined with block reuse during garbage collection causes silent file data loss, a vulnerability likely common to many LFS designs. Our evaluation shows that MC^3 significantly outperforms CrashMonkey in testing efficiency and coverage, enabling the discovery of deep design-level bugs on a standard desktop PC within a practical time frame, thereby providing a powerful and scalable verification framework to enhance the reliability of future storage systems.

Keywords: Model Checking, File System, Crash Consistency, SPIN, F2FS