Paper Critique – Week 12

Name: PHAM CONG VINH Student ID: 2019711010

All File Systems Are Not Created Equal

The context stated in this paper is that nowadays, many important applications are currently implemented on top of modern file system rather than directly accessing on raw disk by themselves. The problem is how we can guarantee an application could be crash-consistent when their dependencies are file systems.

Firstly, the author propose an concept *persistence properties* (the atomicity and the ordering of operations) and develop and tool e Block Order Breaker (BOB) which can explore possible on-disk crash states that may arise by collecting block-level traces underneath a file system and re-orders them. Secondly, they also develop an another novel framework ALICE which allow us analyzing permutations of the system-call trace of workloads in order to clearly present the underlying logic of application, workloads.

The strength of this paper is their argument about how other research can testing the impact of changing persistence properties of file system and by that their solution can become part of the file-system design process.

The limitation of this paper is: Although they experimented their solution with a large number of popular file systems, these file systems are representative only in Linux environment. A comparison with other commercial file systems (NTFS, HFS…) would be more interested.

TxFS: Leveraging File-System Crash Consistency to Provide ACID Transactions

By designing, modern application store their persistent state on multiple files. Although a lot of efforts had been spent into, existing techniques for crash consistency (s logging or using atomic rename) still result in complex protocols and subtle bugs. In this paper, the authors present their intuitive idea, a solution for atomically updating application's persistent state that is Transaction-based updating.

They justify their idea by implementing a transactional file system which exploiting the take advantage of a mature, well-tested piece of functionality in the operating system: the file system journal. This technique is used to ensure atomic updates to the internal state of the file system. The transactional framework allows us to easily implement a number of file-system optimizations (separating ordering from durability, eliminate redundant application IO where temporary files or logs are used to atomically update a file). Finally, they evaluate their solution by modifying SQLite and Git to allow these 2 applications working on their transactional file system – TxFS.

The strength of this paper is that they propose method leveraging fully implemented and tested components of operating system's file system (EXT4) to implement their TxFS. This help saving efforts as well as avoiding error-prone.

The limitation of this paper is, although author stating that their solution support concurrency access, their evaluation on SQLite and Git didn't emphasize on this characteristic. Conducting some experiments on distributed data intensive workload might be more convincing.