Study on Copy-on-Write Persistence

Seri Lee
Seoul National University
sally20921@snu.ac.kr

***Abstract*—Key-value stores are the fundamental components for storing and retrieving data. It is essential to persist data even in the case of a power failure, which could happen at any time. Crash-safe key-value stores are difficult to implement. They maintain large, complex, on-disk data structures that atomically update.**

***Index Terms*—Copy-on-Write, Persistence, Crash Recovery, B-Tree**

I. Introduction

Persistent key-value stores are designed to survive machine failures without losing data. It is difficult to build services that can gracefully handle failure at any time, while still providing good performance.

II. Glossary

A. Shadowing

Shadowing scheme is also known as copy-on-write (CoW) scheme. Shadowing technique is used to ensure atomic update to persistent data structures in file system.

Shadowing means that to update an on-disk page, the entire page is read into memory, modified, and later written to disk at an alternate location. Now all we have to do is to update the pointer in the page table to point to this new page in the disk.

B. Cloning

Clone operation atomically creates a copy-on-write snapshot of a file. It can be referred to as writeable-snapshot, or an extension of the snapshot concept where the snapshot can also be overwritten with new data. Snapshots are supported by many file systems. Clones are a more recent user requirement and are not widely supported yet. Such cloned files are sometimes referred to as reflinks, in light of the associated Linux kernel system calls. By cloining, the file system creates a new inode that shares the same disk blocks as the original file. Thus, this should not be confused with hard links, which are directory entries that associate multiple file names with actual files on a file system.

C. CoW B-tree

Copy-on-Write (CoW) B-tree, it is possible to update the tree just writing in append-only mode. The new root node is always written at the end of the update.

D. Multiversion B-tree

Multiversion B-tree offers the same query/update bounds as the CoW B-tree but only requires asymtotically optimal O(N) space.

E. Snapshots

Snapshots means the read only copy of data set frozen at particular point in time.

III. Versioned Dictionary

IV. Copy-on-Write B-trees

Many file systems use copy-on-write (COW) be-trees to provide atomicity, crash recovery, and efficient clones. When the tree would modify a node, it copies the node to a new location on disk, then modifies the copy. Then the parent must be updated to point to the new location. Changes propagate up to the root of the tree. Nodes that become unreachable will later on be deallocated. If the system crashes, copy-on-write preserves the original tree.

A. ***Paper:*** *"B-trees, Shadowing, and Clones"*

This work describes the first b-tree construction that can coexist with shadowing while providing good concurrency. The gist of the paper's approach is (1) to use top-down b-trees instead of bottom-up b-trees and thus integrate well with a shadowing system (2)remove leaf-chaining (3)use lazy reference-counting for the free space map and thus support many clones.

This author claims that leaf-chaining does not really help range lookups. This is because in order to effectively perform a sequential scan of the b-tree multiple leaf nodes have to be prefetched form disk in parallel. This can be achieved by traversing the tree from the top and issuing prefetch requests for several leaf nodes concurrently. Leaf chaining is of no help for this method of operation.

Shadowing file systems ensure recoverability by taking periodic checkpoints, and logging commands in-between. A checkpoint includes the entire file system tree. Once a checkpoint is successfully written to disk the previous one can be deleted. If a crash occurs the file system goes back to the last complete checkpoint and replays the log. Unreferenced pages will be deallocated.

The process of writing checkpoint is efficient because modifications can be batched and written sequentially to disk. If the system crashes while writing a checkpoint no harm is done, the previous checkpoint remains intact. Command logging is attractive because it combines into a single log-entry a set of possibly complex modifications to the file system. This combination of checkpointing and logging allows an important optimization for the shadow-page primitive.

Before modifying a page, it is "marked dirty". This lets the run-time system know that the page is about to be modified and gives it a chance to shadow the page if necessary.

V. Snapshot

In order to support snapshots, the file system allows having more than a single root. Each root node points to a tree that represents a valid image of the file system. The upshot is that pages can be shared between many snaphots; indeed, whole subtrees can be shared between snapshots.

VI. Conclusion

References

[1]
[2]