FTL with Oracle

Shangda Li
Carnegie Mellon University
Pittsburgh, PA, 15213
shangdal@cs.cmu.edu

Sheng Xu
Carnegie Mellon University
Pittsburgh, PA, 15213
shengx@cs.cmu.edu

Tianjun Ma
Carnegie Mellon University
Pittsburgh, PA, 15213
tianjunm@cs.cmu.edu

ABSTRACT

The flash translation layer (FTL) of an SSD is responsible for mapping logical LBAs to physical pages, and performing garbage collection and wear leveling. A key performance metric of an FTL is Endurance, the number of write requests endured by the FTL until it declares failure. Current state-of-the-art FTL algorithms are not workload-aware and they do not achieve workload-specific optimal page-allocation that can maximize the Endurance of SSDs. We conjecture a superior workload-aware FTL algorithm that can maximize SSD lifetime exists, though devising such an algorithm is difficult and likely involves the use of machine learning. To prove the feasibility of such a workload-aware algorithm, we implement an oracle with complete knowledge of a workload, which can provide hints to the FTL to help the FTL make smarter decisions. We mainly explore two types of hints: various hints to tell the FTL the hotness/coldness of LBAs, and various hints about the optimal internal parameters used by a FTL, such as the block threshold for cleaning. We also implemented an FTL baseline that combines approaches from multiple state-of-the-art FTL algorithms. By showing that the FTL baseline with oracle achieves 20% to 25% higher Endurance than just the FTL baseline, we demonstrate that it is possible for a workload-aware algorithm to outperform state-of-the-art FTL algorithms. We conjecture that a workload-aware algorithm can substitute hints from the oracle with hints deduced on its own, possibly using methods such as machine learning.

1 INTRODUCTION

Background

Solid-state drives (SSDs) based on NAND Flash memory have long been the state-of-the-art non-volatile storage medium. Compared to hard-disk drives, SSDs have faster random I/O speed and can serve multiple requests in parallel. SSDs are widely used in smart phones, personal laptops, data centers, and high-end computing. Internally, an SSD is a collection of physical pages, organized in a hierarchy of packages, dies, planes, and blocks. These layers are hidden to the upper layer, such as file systems and user programs, which view the SSD as a series of Logical Block Addresses (LBAs). Each LBA corresponds to the size of a physical SSD page. Several key characteristic of SSDs are

(1) No overwriting on a SSD page is possible, so every new write must occupy a new page.
(2) Erases can only be done on the granularity of blocks, so valid pages must be migrated before erasing a block.
(3) Each SSD block can only tolerate a limited number of erases, called its lifetime. If a particular block is updated and erased frequently, it will wear out before other blocks and make the SSD inoperable.

Therefore, using direct mapping between LBAs and physical pages is not possible, and SSDs come with a flash translation layer (FTL). The FTL is responsible for address translation (logical-to-physical mapping that maps LBAs to physical pages), garbage collection (erasing blocks to free up space when the SSD is filled up), and wear leveling (load balancing writes across blocks to extend the lifespan of an SSD). Thus the FTL policy for SSDs is very similar to the Segment Cleaning Policy we read about in the Log-Structured File System [1]. Similar to LFS, an FTL cleans with block granularity, migrates pages, and keeps track of page locations.

Motivation

There are numerous varying FTL algorithms. Even though some of them contain dynamically adjusting parameters, e.g. maximum distance between the healthiest and the most worn out block, their core algorithms and the way their parameters adjust is the same across different workloads, where each workload is an online stream of LBA requests sent to the FTL, including READ, WRITE, and TRIM. However, we conjecture that there are different patterns that modern computer programs exhibit in their LBA traces, and for each of these patterns some FTL algorithms perform better than others, and within the same algorithm some combinations of the parameters perform better than others. Such a superior workload-aware algorithm that outperforms state-of-the-art FTL algorithm can extend the lifespan of SSDs, which is desirable given the wide use of SSDs.

However, devising such a workload-aware algorithm is difficult. We speculate that the algorithm is not deterministic and might require machine learning. In this paper, our goal is less ambitious: we aim to show that a superior workload-aware algorithm is feasible. We implement an *oracle*, which knows the entire workload upfront, and uses this complete knowledge to provide *hints* to the FTL. If we show that FTL algorithms can perform better with *hints* from an oracle, then this can serve as a proof-of-concept that there is room for a