SimpleSSD: Modeling Solid State Drives for Holistic System Simulation

Myoungsoo Jung, Jie Zhang, Ahmed Abulila, Miryeong Kwon, Narges Shahidi, John Shalf, Nam Sung Kim, and Mahmut Kandemir

Abstract—Existing solid state drive (SSD) simulators unfortunately lack hardware and/or software architecture models. Consequently, they are far from capturing the critical features of contemporary SSD devices. More importantly, while the performance of modern systems that adopt SSDs can vary based on their numerous internal design parameters and storage-level configurations, a full system simulation with traditional SSD models often requires unreasonably long runtimes and excessive computational resources. In this work, we propose `SimpleSSD`, a high-fidelity simulator that models all detailed characteristics of hardware and software, while simplifying the nondescript features of storage internals. In contrast to existing SSD simulators, `SimpleSSD` can easily be integrated into publicly-available full system simulators. In addition, it can accommodate a complete storage stack and evaluate the performance of SSDs along with diverse memory technologies and microarchitectures. Thus, it facilitates simulations that explore the full design space at different levels of system abstraction.

Index Terms—Hardware, computer architecture, parallel processing, computational modeling, systems simulation, microprocessors, software

1 Introduction

In the past decade, solid state disks (SSDs) have reshaped modern memory hierarchy by replacing conventional spinning disks and/or blurring the boundary between main memory and storage systems. Thanks to their high performance and low power consumption characteristics, SSDs have already become the dominant storage type in diverse computing domains, ranging from embedded to general-purpose and high-performance computing systems. This in turn has led to a wide spectrum of research, including the comprehensive exploration of the full design space, storage stack optimization, and architecture renovation at various layers of memory and storage subsystems.

While simulations are indispensable for system designers and computer architects, very few SSD simulators have been released to the public domain [5], [7], [8], [10]. Further, these simulators have constraints that prevent them from filling the needs of design space exploration for emerging memory and storage subsystems. First, all existing SSD simulators lack system-level simulation capability, and integrating these simulators with publicly-available full-system simulators is a non-trivial task. While the execution of a CPU instruction only takes a few cycles in a simulation, a storage access requires tens of millions (even billions) of cycles for its service. Similarly, a file access in an accurate SSD simulation model can exhibit a long execution time because it needs to go through the SSD's intricate software stack and hardware architecture. Traditional SSD simulators cannot fully account for the important functionalities of the underlying firmware and model the underlying hardware in detail. Thus, they are far from capturing the critical features of contemporary high-performance SSD architectures.

In this work, we propose `SimpleSSD`, a high-fidelity simulator that models all of the detailed characteristics of hardware and software while simplifying the nondescript features of storage internals such as multi-cycle operations to address a target page on a flash interface. The proposed hardware and software simplifications allow `SimpleSSD` to accommodate a complete storage stack. Thus, system designers and computer architects can evaluate the SSDs performance along with diverse memory technologies and can explore the full design space of an SSD architecture. Moreover, `SimpleSSD` can easily be integrated with publicly-available full-system simulators and can capture relevant CPU performance characteristics impacted by different storage types employed by the system. As a case study, we integrated `SimpleSSD` with the popular full-system simulator, gem5 [6], and evaluated its system-level performance from various aspects. Note that traditional SSD simulators [5], [7], [10] capture only storage-related metrics such as bandwidth and latency by replaying block-level I/O traces; this ignores system-level interaction between the host-side CPU and storage subsystems. In contrast, the proposed `SimpleSSD`[1] can report detailed information from low-level memory to each firmware module in order to determine the host-side CPU performance while executing entire applications.

2 SSD-Enabled System Simulation Overview

Fig. 1 shows an overview of a holistic system simulation with the proposed `SimpleSSD`. Application(s) simulated on the host can place an I/O request through a virtual file system (VFS) and native file system. The VFS buffers small-sized requests through a page cache, whereas the native file system manages the data accesses and system memory. The request then arrives at a block layer that reorders and combines multiple requests into a specific order. This CPU processing part can communicate with the layered firmware of `SimpleSSD` via a disk controller. Then, the layered firmware simulates the SSD process part by interacting with an abstraction model, which simulates the given SSD hardware architecture including multiple flash dies, module interfaces, and channels. Although `SimpleSSD` leveraged gem5 running in full-system mode to simulate such CPU processing in this study, it can easily be integrated into other full-system simulators such as MARSSx86 [13].

Layered Firmware. One of the main challenges of simulating an SSD is supporting diverse flash firmware versions, which greatly influences the target storage performance. We model a flexible *flash translation layer* (FTL) whose address translation mechanism can simply be reconfigured based on different associativity granularities defined by system architects. We also decouple I/O scheduling and page allocation mechanisms from the FTL so that new scheduling proposals that are aware of SSD-internal parallelism can be embedded without changing the FTL. Although we do not cover all types of potential FTLs, the implemented reconfigurable mapping algorithm can capture/support diverse operational characteristics of a block-level mapping FTL, a fully-associative FTL, and various hybrid mapping schemes that employ different levels of block and page mapping tables in their address translations. In addition, our simplified but reconfigurable layered firmware also offers diverse research opportunities where system and computer architects can simply modify some performance-critical components such as garbage collection and wear-leveling algorithms with different mapping mechanisms.

- *M. Jung, J. Zhang, and M. Kwon are with the Computer Architecture and Memory Systems Lab, Yonsei University, Seoul 03722, Republic of Korea. E-mail: {m.jung, jie}@yonsei.ac.kr, mkwon@camelab.org*
- *A. Abulila and N. Sung Kim are with the University of Illinois Urbana-Champaign, Champaign, IL 61820. E-mail: {abulila2, nskim}@illinois.edu*
- *N. Shahidi and M. Kandemir are with Pennsylvania State University, State College, PA 16801. E-mail: nxs314@psu.edu, kandemir@cse.psu.edu*
- *J. Shalf is with the Lawrence Berkeley National Laboratory, Berkeley, CA 94720. E-mail: jshalf@lbl.gov*

Manuscript received 22 Feb. 2017; revised 20 Mar. 2017; accepted 12 May 2017. Date of publication 10 Sept. 2017; date of current version 19 Mar. 2018.

(Corresponding author: Myoungsoo Jung.)

For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/LCA.2017.2730658

1. The SimpleSSD source code can be freely downloaded from the following website: http://simplessd.camelab.org.