Enhancing FlashSim Simulations for High Performance Computing Storage Systems

Omar Rodriguez* (Undergraduate, Junior), Wei Xie and Yong Chen[a] (Advisors)
*Department of Electrical & Computer Engineering and Computer Science
Polytechnic University of Puerto Rico
San Juan, PR 00918
Email: rodriguez_80863@students.pupr.edu
[a]Department of Computer Science
Texas Tech University
Lubbock, TX 79413
Email: wei.xie, yong.chen@ttu.edu

***Abstract*-This work describes the design and development of an enhanced FlashSim simulator framework and toolkit. The FlashSim is one of the most popular SSD simulators. SSD simulator focuses on software components, specifically FTL (flash translation layer) schemes, garbage collection, and wear-leveling policy. In order to simulate and analyze the performance results, the simulator needs block I/O traces. This work extends the FlashSim simulator with leveraging the blktrace tool to automate collecting traces from different application workloads. We also develop a converter that transforms the I/O traces into the format needed for the FlashSim simulator. The proposed extension enhances the FlashSim simulator and automates trace collection, conversion, and analysis.**

I. Introduction

NAND Flash based Solid State Drives (SSDs) have increasing demands on high-end/high-performance computing systems. Unlike the traditional rotating architecture of HDD's, SSD characteristics include free of mechanical components, high bandwidth, low latency, shock resistance, and low power consumption.

With the SSDs usage rapidly increasing, most of the design details, however, are not fully revealed to the public domain. The research of SSDs have been dominated by using simulators, such as FlashSim. To provide stimulus to the simulator and analyze the SSD performances, the block I/O traces are needed. The blktrace is a block layer I/O tracing mechanism which provides detailed information about request queue operations [3].

In this work, we extend the FlashSim with automating trace collection via using the blktrace tool for application/-benchmark workloads including online streaming workload. We used the IOzone benchmark to produce different workloads specifically for evaluation. The video stream traces were collected for evaluation as well and for studying the impact of I/O traffic. The collected traces were translated with a converter into the FlashSim format in order to simulate and analyze new SSD design performance.

II. Design and Experiment

A. Design

Due to the lack of easy data manipulation of the I/O traces, a simple design solution to obtain the data was devised. The program allows for data acquisition from the blktrace tool and enables Flashsim to perform the appropriate simulations. Since the data obtained locally from blktrace is not ready to be fed into FlashSim, another tool was designed to convert the necessary information into a format that FlashSim can use. The architecture of our design is shown in Figure 1.

B. Experiment

The experiments were conducted in the Data-Intensive Scalable Computer Laboratory at Texas Tech University. We used the IOzone file system benchmark and the online streaming to produce different workloads. Simultaneously, the I/O traffic of the workloads were traced using blktrace tool extension for FlashSim. The output file obtained from the blktrace was automatically translated in order to convert the file in the format for FlashSim. After the conversion, the FlashSim simulator was used to analyze and study the behavior of different workloads, as well as to study different designs of SSDs, e.g. FTL algorithms, wear leveling algorithms, garbage collection policy, etc.

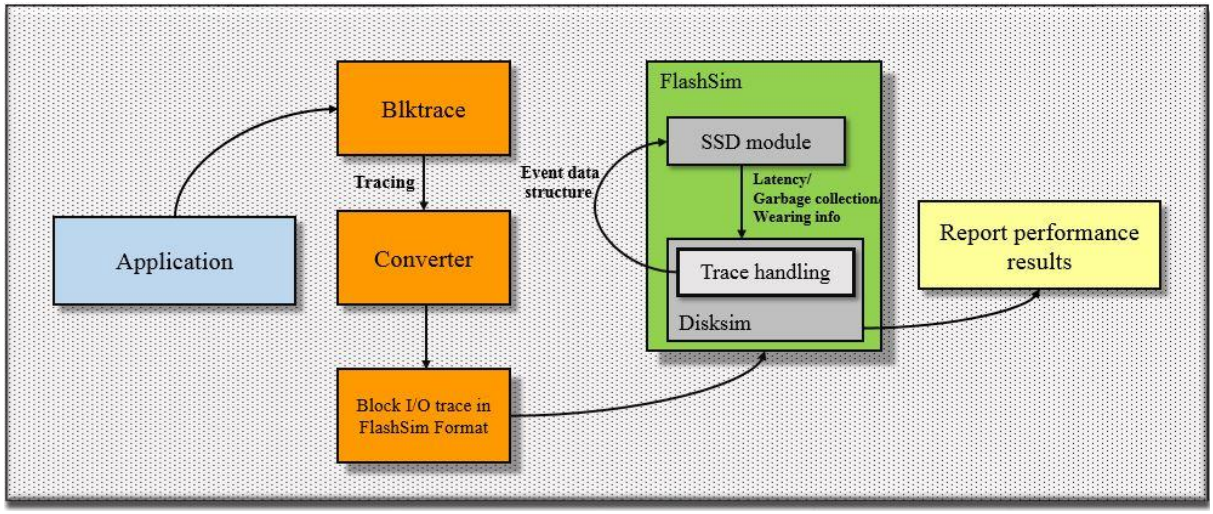

Figure 1: Design Architecture to Enhance the FlashSim Simulator

III. Results and Analyses

We conducted experiments with the IOzone benchmark. Two different workload patterns, including sequential write and random write; each of these patterns was traced with different block sizes: 512 and 1024 Bytes. In addition, a trace from an online video streaming application was also collected. All the traces were converted and fed into the FlashSim simulator. In the simulation, we configured the simulator to run with different FTL algorithms: DFTL [4], FAST [5], and page mapping scheme [1].

As shown in Figure 2, we can see that, for the sequential write pattern, both the FAST and DFTL perform inferior than the page mapping scheme. However, the performance gap between them is less significant in the sequential pattern than the random pattern for the DFTL. The reason is that the sequential workload introduces less address translation overhead than the random workload. But for the FAST, this is due to that the random workload incurs more expensive merge operation.

In Figure 3, we plot the response time of different FTLs with the streaming application. We can observe a huge difference in the average response time between the DFTL or FAST and the page mapping scheme. The online streaming writes a significant amount of data into the disk. The overhead incurred in the DFTL and the FAST increases the wait time and thus makes the contention in the device even more severe.

Using the example traces described above, we demonstrated that our proposed enhanced SSD simulation tool is capable of combining the job of block IO tracing and FTL level SSD simulation together. This tool shows great feasibility of understanding performance characteristics of the workload pattern of the applications and the design choices of the FTL algorithms. It would benefit the researchers in both the workload analysis and the storage optimization area.

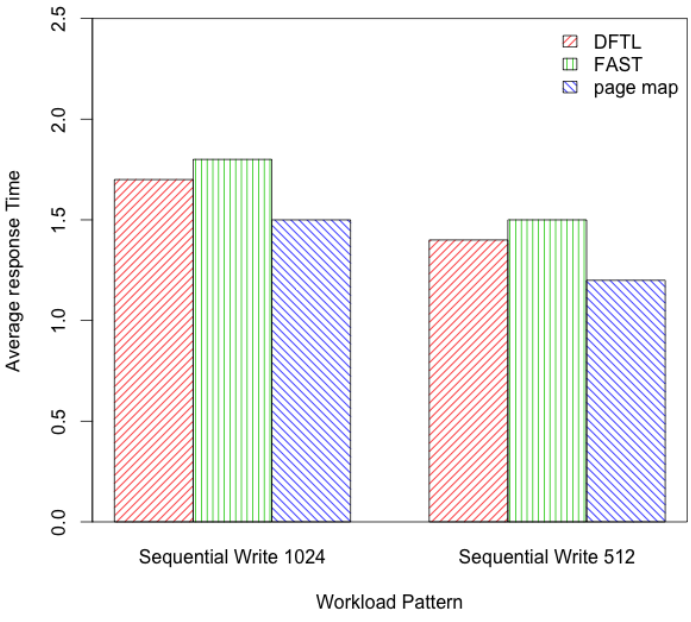

Figure 2: The response time simulation result of IOzone sequential write workload trace.

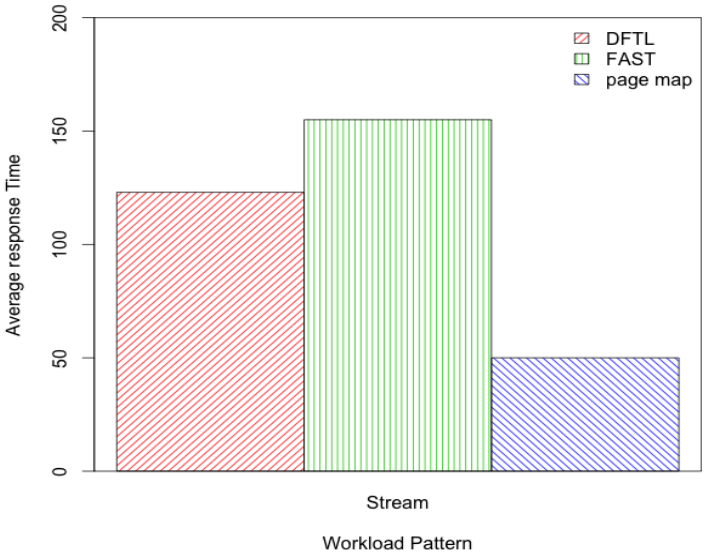

Figure 3: The response time simulation result of Streaming media workload traces

IV. Conclusion and Future Work

In this work, an enhanced FlashSim simulator framework and toolkit was designed and prototyped. The enhancement focuses on automating trace collection and conversion for the FlashSim simulations. The automatic trace collection leverages the blktrace utility to collect block I/O traffics. A converter was developed to translate traces for the FlashSim format. The designed FlashSim extension provides with ease of access to simulations with desired workloads. Both benchmark and online streaming application workloads were evaluated to verify the feasibility. This study enhances simulation functionality and promotes research on flash memory based storage devices. Future research and development include detailed filter/analysis utility on traces.

Acknowledgement

This research is supported in part by the Edward Whitacre Jr. College of Engineering undergraduate research program and the National Science Foundation under grant CNS-1162488.

References

[1] Chung, T. S., Park, D. J., Park, S., Lee, D. H., Lee, S. W., & Song, H. J. (2009). A survey of flash translation layer. Journal of Systems Architecture, 55(5), 332-343.

[2] Kim, Y., Tauras, B., Gupta, A., & Urgaonkar, B. (2009, September). Flashsim: A simulator for nand flash-based solid-state drives. In Advances in System Simulation, 2009. SIMUL'09. First International Conference on (pp. 125-131). IEEE.R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[3] "blktrace User Guide" http://www.cse.unsw.edu.au/~aaronc/iosched/doc/blktrace.html

[4] Gupta, A., Kim, Y., & Urgaonkar, B. (2009). DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings (Vol. 44, No. 3, pp. 229-240). ACM

[5] Lee, S. W., Park, D. J., Chung, T. S., Lee, D. H., Park, S., & Song, H. J. (2007). A log buffer-based flash translation layer using fully-associative sector translation. ACM Transactions on Embedded Computing Systems (TECS), 6(3), 18.