Performance Impact and Interplay of SSD Parallelism through Advanced Commands, Allocation Strategy and Data Granularity

Anshu Aviral
BITS Pilani, K K Birla Goa Campus
aviral.2815@gmail.com

PROBLEM STATEMENT

The advent of NAND based Solid State Devices (SSD) has revolutionized the storage industry, with its diverse application, running the gamut from low end PC's to high end supercomputer servers. This experimental paper focuses on the following SSD related issues:

- Impact of various features of flash memory, like flash page size, allocation schemes, advanced commands and priority order of SSD parallelisms, on its performance and endurance has not been studied extensively. These internal features and their interplay proves to be pivotal in getting the best out of SSD.

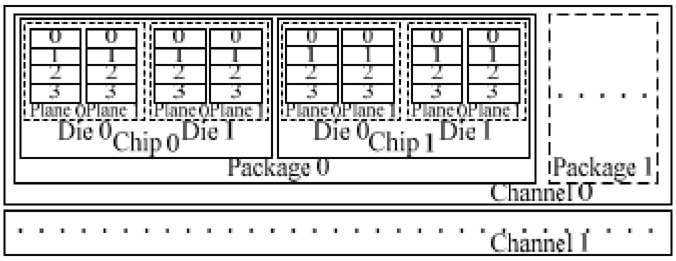
- Lack of an open-source and a high accuracy SSD simulator that adheres to the restrictions imposed by various levels of parallelism, and to provide a scheme to compare the simulation measurements with a real SSD system, proves to be a bottleneck for researchers.

BACKGROUND

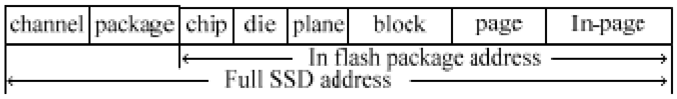
NAND based flash memory supports block level random access in contrast to NOR based flash memory. A Single Level Cell (SLC) is a flash technology that stores one bit per cell while Multi Level Cell (MLC) can store multiple bits per cell.

Flash Memory Architecture

The terms associated with the architecture of flash memory are *channel*, *package*, *chip*, *die*, *plane* and



block. The given diagram illustrates the hierarchical strucutre of each of these components. A flash block consists of either 64 or 128 pages which are subdivided into various 512B sub-pages, which further subsumes error correction codes (ECC), logical page number and sub-page state. An address is seprarated into the following eight segments.



Flash Commands

A *read* operation fetches data from the target page. A *write/program* operation writes data to a target page. An *erase* operation resets all bits of a target block to 1. Three advanced commands are also provided to boost SSD performance. *Copy-back* command moves one page of data from one page to another (both odd or both even) in the same plane, without occupying the I/O bus. *Multi-plane* command executes multiple erase, program or read operation in the time of single operation execution. The i*nterleave* command executes several page read, page write, block erase and multi-plane read/write/erase operations in different dies of the same chip simultaneously.

Allocation Schemes

The choice of free physical pages to accomodate the logical pages to be written to SSD is determined by its allocation strategies. *Static allocation* assigns a logical page to pre-determined channel, package, chip die and plane, calculated by some formula. Only then it assigns it to a free physical page. *Dynamic allocation* assigns a logical page to any free physical page of the entire SSD based on factors like idle/busy state of the channel, the idle/busy state of chips, the erasure count of blocks, the priority order of parallelism and so on.

There are four levels of parallelism in SSD, namely channel level, chip level, die level and plane level. The performance impact of exploiting these levels are examined in the following sections.

Before moving ahead let's look at three restrictions that must be adhered to, when executing flash commands -

a) Within a block, the pages must be programmed consecutively in increasing order of page address.
b) The addresses of the source page and destination page must be both odd or both even in copy-back.
c) The blocks executing a multi-plane erase operation must have the same chip, die and block addresses.

The SSD simulator

An SSD simulator (called SSDsim) was designed and implemented to overcome the issue stated in the problem statement. It is an event-driven, three-tiered and modularly structured single threaded program written in C. The three-tiered SSDsim design consists of the buffer and request-scheduling module at the top, the FTL and allocation module in the middle, and the low-level hardware platform module at the bottom.

THE SIMULATION RESULTS

A set of real world workloads were to used for simulation. These included data from large financial institutions, an Internet web search machine, Microsoft's several live file servers, Microsoft Exchange 2007 and a few more such sources. The paper presents the evaluation of the following parameters which have notable impact on SSD performance.

Flash Page Size

First look at the flash page size facilitates larger page size as larger blocks of data getting transferred at once will decrease response time. But let's consider the following factor. An *uncovered* update operation requires one more flash read operation than a a *covered* update operation. Depending upon the nature of the workload, more uncovered operation is ought to occur if the page size in large. Hence, the request's response time can increase as the size of a flash page increases.

Allocation Schemes

As per the experiments performed by the SSDsim, the paper concludes that static allocation scheme outperforms dynamic allocation schemes in serving read requests. In an aged SSD, dynamic allocation scheme should be preferred with only exception being read-dominant workloads. Also, dynamic scheme is better than static scheme when considering wear-levelling performance.

Advanced Commands

The advanced commands can either be used *blindly* or *wisely.* Using the copy-back command wisely, when the source page and the destination page has different parity can cause reduction in I/O performance and endurance of SSD. Using multi-plane write command blindly improves I/O performance but reduces endurance under most workloads. Multi-plane read proves to be insignificant except in read-dominant workloads, where it increases I/O performance. Interleave improves I/O performance without degrading endurance. Thus, it should be used profusely.

Priority Order of Parallelism

The priority order of parallelisms in SSD as shown experimentally, should be, in the given order -
(1) channel level parallelism
(2) die level parallelism
(3) plane level parallelism
(4) chip level parallelism

CONCLUSION

Having looked at the role of each parameter on the performance and endurance of SSDs, let's look at the interplay of these parameters and how they improve or degrade the performance. Based on the individual analysis, the recommended approaches to using the advanced commands are -

1) Use the copy-back command, the Multi-plane write command, and the interleave command wisely under all circumstances

2) Use the copy-back command wisely, the Multi-plane write command blindly, and the interleave command ubiquitously

As a final conclusion, the combined use of advanced commands based on Approach (2) achieves the best performance but leads to SSD endurance degradation, while Approach (1) achieves the less performance gain but without any endurance loss.