Unleashing the Potentials of Dynamism for Page Allocation Strategies in SSDs

Arash Tavakkol[†], Mohammad Arjomand[†], and Hamid Sarbazi-Azad[†‡]
[†]HPCAN Lab, Computer Engineering Department, Sharif University of Technology, Tehran, Iran
[‡]School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran
{tavakkol,arjomand}@ce.sharif.edu, azad@{sharif.edu,ipm.ir}

ABSTRACT

In Solid-State Drives (SSDs) with tens of flash chips and highly parallel architecture, we can speed up I/O operations by well-utilizing resources during page allocation. Proposals already exist for using static page allocation which does not balance the IO load and its efficiency depends on access address patterns. To our best knowledge, there have been no research thus far to show what happens if one or more internal resources can be freely allocated regardless of the request address. This paper explores the possibility of using different degrees of dynamism in page allocation and identifies key design opportunities that they present to improve SSD's characteristics.

Categories and Subject Descriptors

D.4.2 [**Operating Systems**]: Storage Management—*Secondary Storage*

Keywords

Solid State Drive, Page Allocation, Dynamism.

1. INTRODUCTION

The main feature of modern SSDs is that they employ a large set of various resources, including flash packages and communication channels that are organized to achieve high level of internal parallelism. Besides, the architecture of flash packages provides increased number of planes and dies to boost parallelism. Each of these parallelism levels has its own limitations and opportunities that causes SSD's performance to be highly dependent to the strategy used for conducting flash operations in parallel. This is known as *allocation strategy* and determines how to efficiently exploit resource parallelism and avoid their inherent limitations. The main focus of past studies was on allocation schemes where the target of a flash operation is statically determined using a predefined priority order of parallelism levels [1, 3]. The efficiency of these approaches highly depends on the address patterns of I/O requests and may lead to conflicts on shared resources. To address this issue, one can relax deterministic resource assignment in one or more levels of parallelism, and use dynamism instead. Accordingly, we introduce new allocation schemes with various degree of freedom of one, two, three or four, and analyze their performance issues with respect to static allocation. While a number of prior work adopted fully-dynamic allocation (with a freedom degree of 4) as the extreme design point [1], our evaluations confirm that it cannot efficiently utilize all parallelism opportunities. Especially, allocations with freedom degrees of 2 and 3 can better exploit intra-flash parallelism.

SIGMETRICS'14, June 16–20, 2014, Austin, Texas, USA.
ACM 978-1-4503-2789-3/14/06.
http://dx.doi.org/10.1145/2591971.2592013.

2. DYNAMIC ALLOCATION STRATEGIES

The page allocation (*PLAlloc*) determines the target Channel ID (CID), Way ID (WID), Die ID (DID), and Plane ID (PID) of a page-sized transaction following a specific priority order of the parallelism levels. Unfortunately, making an appropriate decision for *PLAlloc* prioritization strategy is not a trivial task since each choice acts in favor of some parallelism levels and diminishes the chance of utilizing others. Worse, some of the parallelisms, such as plane-level, are inherently restricted [1] and require extra design considerations to achieve reasonable performance [2]. In static strategies, CID, WID, DID and PID are calculated by successive divisions of the LPA. This approach makes their performance to be highly dependent to the LPA access patterns. As an alternative approach, we suggest to relax deterministic assignment of resource ID(s) in one or more dimensions of parallelism and use dynamic assignment instead. For a certain level of parallelism, dynamism may be provided by *round-robin* or *load-aware* resource allocations. Our choice is busy-aware round-robin which immediately tries to assign next resource, if current candidate is busy. This results in a better address striping and resource utilization especially when resources are limited. Based on this definition, 41 dynamic *PLAlloc* strategies are possible which are classified into four categories by considering their degree of freedom:

Degree one. Just one resource ID is determined dynamically and others are assigned in the static approach:

$$L_1L_2L_3, \forall i, L_i \in \{C, W, D, P\} \quad for \quad L_i \neq L_j, \quad i \neq j.$$

Degree two. Two resource IDs are determined dynamically and the others are extracted statically:

$$L_1L_2, \forall i, L_i \in \{C, W, D, P\} \quad for \quad L_1 \neq L_2$$

Degree three. One resource ID is statically calculated:

$$L_1, L_1 \in \{C, W, D, P\}$$

Degree four. There is one *PLAlloc* strategy, namely F, to which all resource IDs are dynamically assigned. Various implementations of this strategy were previously presented [1, 6] as the only choice of dynamic allocation.

Figure 1 presents the steps of translation within CP dynamic allocation. As can be seen, CID and PID are deterministically calculated from the LPA, and the two remaining resource IDs are allocated using busy-aware round-robin algorithm. Note that when more than one level of parallelism is dynamically assigned, *PLAlloc* calculates them in order of channel, way, die, and plane for better striping.

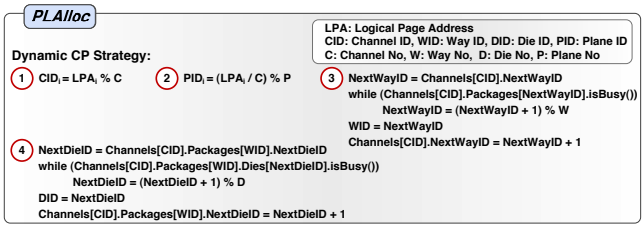
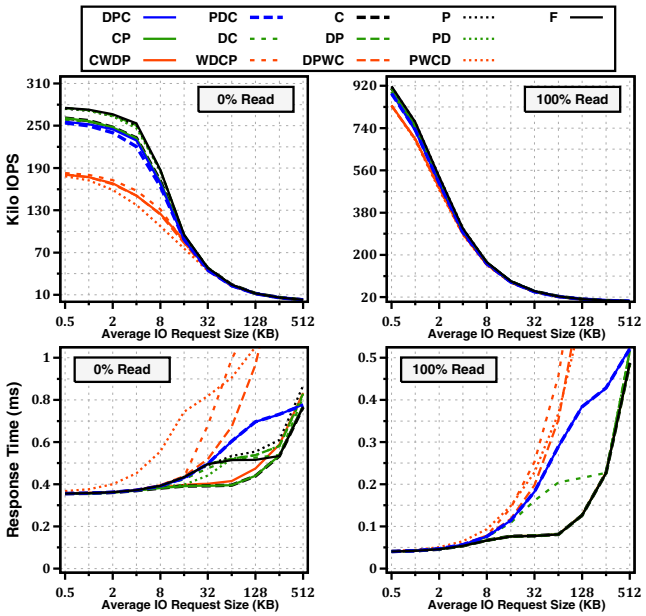


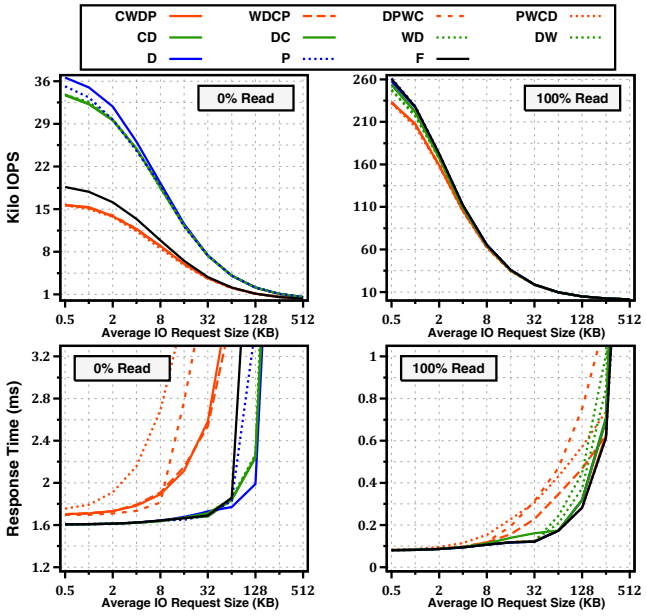
Figure 1: The steps of translation for dynamic CP *PLAlloc*.

3. EVALUATION

We perform discrete-event simulations using SSDSim [1]. We model two SSD configurations, SSD-ENT and SSD-CLN, representing the enterprise and client class SSDs, respectively. SSD-ENT uses 64 SLC NAND flash chips [5] organized in 8 channels while SSD-CLN has 16 MLC NAND flash chips [4] connected through 4 communication channels. We conducted a set of micro-benchmark simulations with varying I/O request sizes and read/write ratios. The evaluations are performed on *PLAllocs* that present best performance based on our observations. Figure 2a depicts the results of IOPS and RT for SSD-ENT. As expected, dynamism greatly enhances the IOPS of small and medium size requests and F, PD, and DP always show the outstanding results among all possible choices. Besides, increasing the request size weakens the superiority of dynamic schemes since the probability of unresolvable resource contentions grows. As the final point, the RT results prove that the amount of resources within SSD-ENT are enough to equally decrease the contentions probability for both static and dynamic schemes. Regarding this metric, the C, CP, and CWDP schemes show the best results while F and PD are ranked second. As can be seen in Figure 2b, the micro-benchmark results for SSD-CLN show the impressive performance impact of dynamic schemes, especially when using D strategy. In fact, the IOPS results of D are considerably higher than static *PLAlloc*s and it even shows better values with respect to other dynamic schemes. Regarding RT results, D shows minor increase for medium size requests but achieves noticeable improvement for larger sizes. Our observations show that the superior performance of D is due to higher chance of exploiting multi-plane and multi-die parallelism in this scheme. We deduce that either of F and PD are the best *PLAlloc*s for SSD-ENT and D is the proper solution for SSD-CLN.



(a)



(b)

Figure 2: Performance analysis of sample *PLAlloc* strategies for (a) SSD-ENT and (b) SSD-CLN.

4. REFERENCES

[1] Y. Hu et al. Performance impact and interplay of ssd parallelism through advanced commands, allocation strategy and data granularity. In *ICS '11*.

[2] M. Jung et al. Physically addressed queueing (PAQ): improving parallelism in solid state disks. In *ISCA'12*.

[3] M. Jung and M. Kandemir. An evaluation of different page allocation strategies on high-speed ssds. In *HotStorage'12*.

[4] Micron Technology, Inc. MT29E256G08CMCAB NAND flash memory, 2010.

[5] Micron Technology, Inc. MT29F128G08AMCABJ2 NAND flash memory, 2010.

[6] C. Park et al. Exploiting internal parallelism of flash-based ssds. *Computer Architecture Letters*, 9(1):9–12, 2010.