Title: Parallel all the time: Plane Level Parallelism Exploration for High Performance SSDs

Name: Congming Gao **Email:** gaocm92@gmail.com

Advisor: Dr. Liang Shi, Chun Jason Xue **Estimated graduation date:** July, 2020

Institution: College of Computer Science, Chongqing University, Chongqing, China

Whether the work has been presented at previous forums: No.

This work has been accepted by Massive Storage Systems and Technology (MSST) 2019.

Dissertation:

Solid state drives (SSDs) are constructed with multiple level parallel organization, including channels, chips, dies and planes. Among these parallel levels, plane level parallelism has the most strict restrictions: for two operations that can be issued simultaneously to two different planes, they not only need to be of the same type (i.e., read or write) but also need to have the same in-plane address, making it challenging to explore [1][2][4].

First, a naive solution to address the above issues is to write data at the aligned points greedily. However, if the current write points are unaligned, writing data at the aligned points lead to wasted space. As shown in Figure 1(a), in Figure 1(a)-(1), the current write points are unaligned. Traditionally, if two write operations, *W1* and *W2*, are issued to the two planes in the same die, they will be processed sequentially. If they are written to the aligned pages, a free page in Plane 1 would be wasted, as shown in Figure 1(a)-(2). Second, internal SSD activities, e.g., GC, also face such a problem: in Figure 1(b), after moving valid pages in each plane in Figure 1(b)-(1), the new write points (*WP* in the figure) become unaligned, as shown in Figure 1(b)-(2).

In order to maximize the access performance, a *from plane to die* optimization framework (called as SPD) is proposed to exploit the plane level parallelism through smartly satisfying the strict restrictions *all the time*.

Basically, SPD takes the following strategies to achieve the objective, as shown in Figure 2. SPD adds two new components: a die level write construction and a die level GC. The die level write construction is designed to maintain aligned write points for host writes. The die level GC is designed to maintain aligned write points for GC induced page movement.

For die level write construction, SPD exploits the SSD buffer to choose *N* dirty pages and writes them back to one die simultaneously. This helps to convert one die access to *N* page writes at the aligned in-plane address. This is referred to as *Die-Write*. Similarly, the read access to the die is referred to as *Die-Read*. For die level GC, it is activated at the multiple planes in a die at the same time. In addition, all writes induced from the valid page movements is processed in the unit of *N* page writes to maintain the aligned write point. This is referred to as *Die-GC*.

SPD exploits SSD buffer to assist die level write construction. An SSD buffer evicts a multiple of *N* dirty pages from one die at a time such that these pages can be written using *Die-Write*. To construct same in-plane address, *Die-Write* adopts dynamic allocation at plane level for constructing aligned write points of all planes.

For *Die-GC*, the goal is to speed up the GC process with minimal GC cost. For this purpose, SPD activates GC at all the planes in the same die at the same time with carefully selected victim blocks. By adopting *Die-Write* instead of sequential page writes, SPD improves reclaim effectiveness by reducing the most timing cost.

We evaluate the proposed SPD using a significantly extended SSD simulator (SSDSim) and compare it to the state-of-the-arts. The experimental results show that SPD is able to significantly improve write performance of SSDs without read performance impact.

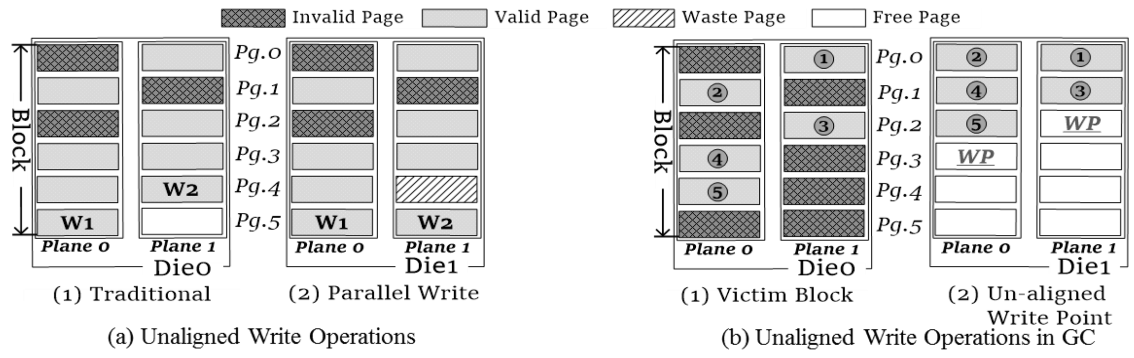

Figure 1. The problems of unaligned write operations.

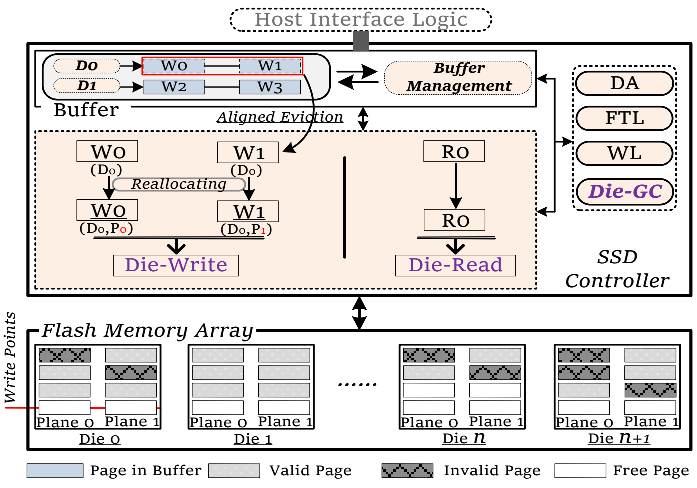

Figure 2. The overview of proposed framework.

Reference and my Paper list:

[1] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and S. Zhang. Performance impact and interplay of ssd parallelism through advanced commands, allocation strategy and data granularity. In ICS, 2011.

[2] **Congming Gao,** L. Shi, M. Zhao, C. J. Xue, K. Wu, and E. Sha. "Exploiting Parallelism in IO Scheduling for Access Conflict Minimization in Flash-based Solid State Drives". In MSST, 2014.

[3] **Congming Gao,** L. Shi, K. Wu, C. J. Xue, and E. Sha. "Exploit Asymmetric Error Rates of Cell States to Improve the Performance of Flash Memory Storage Systems". In ICCD, 2014.

[4] **Congming Gao**, L. Shi, C. Ji, Y. Di, K. Wu, C. J. Xue and E. Sha. "Exploiting Parallelism for Access Conflict Minimization in Flash-based Solid State Drives". In IEEE TCAD, 2018.

[5] **Congming Gao**, L. Shi, Y. Di, Q. Li, C. J. Xue, K. Wu and E. Sha. "Exploiting Chip Idleness for Minimizing Garbage Collection Induced Chip Access Conflict on SSDs". In ACM TODAES, 2018.

[6] **Congming Gao,** L. Shi, Y. Di, Q. Li, C. J. Xue and E. Sha. "An Efficient Cache Management Scheme for Capacitor Equipped Solid State Drives". In GLSVLSI, 2018.

[7] **Congming Gao,** Y. Di, A. Deng, D. Liu, C. Ji, C. J. Xue and L. Shi. “Flash Friendly File System Aware Mapping Cache Design on Solid State Drives”. In NVMSA, 2018.