ICIC Express Letters
Volume **10**, Number **3**, March **2016**
ICIC International ©2016 ISSN 1881-803X
pp. **691–697**

PERFORMANCE CONSIDERATIONS FOR WRITING DATA TO SOLID-STATE DRIVES

Shih-Yu Liu, Dereje Tekilu Aseffa and Chin-Hsien Wu

Department of Electronic and Computer Engineering
National Taiwan University of Science and Technology
No. 43, Sec. 4, Keelung Rd., Da'an Dist., Taipei City 106, Taiwan
{ M9902135; chwu }@mail.ntust.edu.tw; dereteklu@gmail.com

Received November 2015; accepted January 2016

Abstract. *In recent years, solid-state drives (SSDs) which use NAND flash memory have become popular and have replaced traditional hard-disk drives in some applications. Due to the characteristics of NAND flash memory, the management activities inside SSDs could degrade the access performance. In the paper, we will analyze and consider the performance for writing data to SSDs. The objective of the paper is to achieve a moderate decline in overall speed for SSD-based storage systems.*
Keywords: Non-volatile storage systems, Solid-state drives, Performance

1. **Introduction.** In recent years, solid-state drives (SSDs) which use NAND flash memory have become popular and have replaced traditional hard-disk drives in some applications. However, users cannot directly overwrite data in NAND flash memory. We must write the new data in a new location and then invalidate the old data due to the characteristics of NAND flash memory. The invalid data will be recycled by the activities of garbage collection inside SSDs. Therefore, performance considerations for SSDs must be investigated to improve the overall access speed and lifetime of SSDs. Based on the observations on SSDs, large amounts of written data could consume more free space and also cause frequent activities of garbage collection to degrade writing speed. Therefore, we must consider the information of written data and storage devices, and effectively distribute data with a moderate decline in overall speed for SSDs. We list the major contributions of the paper in the following.

- Three SSDs are used to conduct our investigations: Intel 320 40G SSD, Intel 320 160G SSD, and ADATA S599 40G SSD. Two experiments, that is, "changing the size of the write request" and "changing the update ratio of the written data", were performed for each SSD under actual workloads and simulated workloads.
- We propose two distribution rules for append data and update data, respectively. Based on our observations, the distribution rules must consider (1) the writing speed of SSDs, (2) the size of the write request, (3) the update ratio of the written data (i.e., hot/cold data), and (4) the declining speed ratio of SSDs.

The rest of this paper is organized as follows. Section 2 is the related work and motivation. Section 3 proposes performance considerations for writing data to solid-state drives. Finally, Section 4 is the conclusion.

2. **Related Work and Motivation.** Many previous studies [1-7] have used NAND flash memory in different storage systems to improve the storage performance. Different from the previous work, the proposed method will consider (1) the writing speed of SSDs, (2) the size of the write request, (3) the update ratio of the written data, and (4) the declining speed ratio of SSDs, when large amounts of data are stored in SSDs. Furthermore, the

previous work did not differentiate between append data and update data, which may result in an uneven distribution of data in SSDs and reduce the overall speed. Furthermore, because we could buy different models of SSDs at different times, it is possible to install them at the same time. From the economic point of view, we should continue to use them until they are damaged. Especially when the hardware resources (e.g., CPU, main memory, and storage devices) are abstracted into virtual resources for virtual machines, we should integrate the different models of SSDs into a logical view of storage device. Therefore, the objective of the paper is to investigate and consider the performance of SSDs when they are integrated into storage systems.

3. Performance Considerations for Writing Data to Solid-State Drives.

3.1. **Overview.** As shown in Figure 1, the proposed method can be adopted in a logical view of storage device that consists of N different storages (e.g., SSDs), where the device mapping is a mechanism to integrate and manage all the storages. When the written data will enter the logical view of storage device, the written data can be distinguished between the append data and the update data. If the written data are the append data or the update data, two distribution rules can be applied to the corresponding data, respectively. Using the two distribution rules, high-performance SSDs can be achieved.

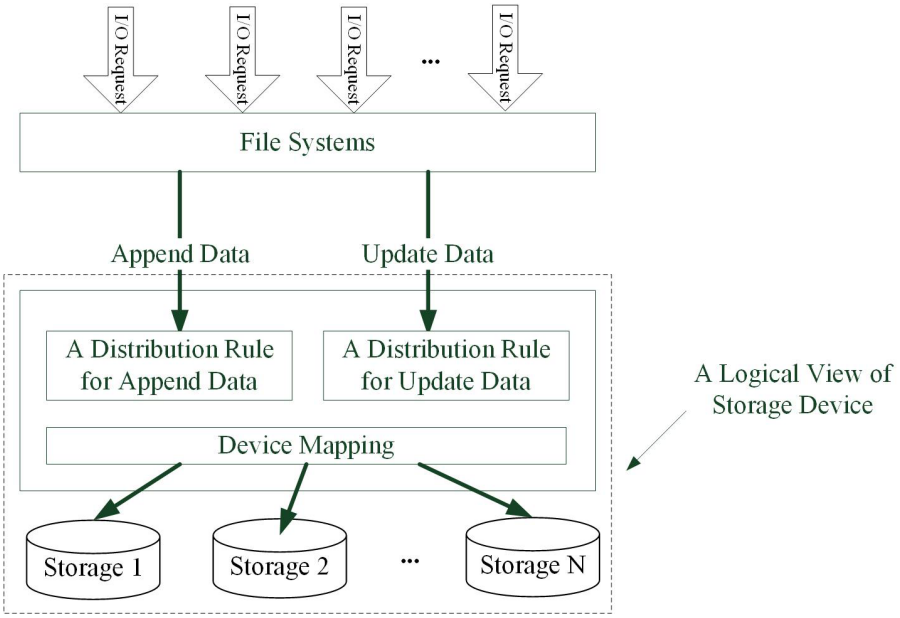
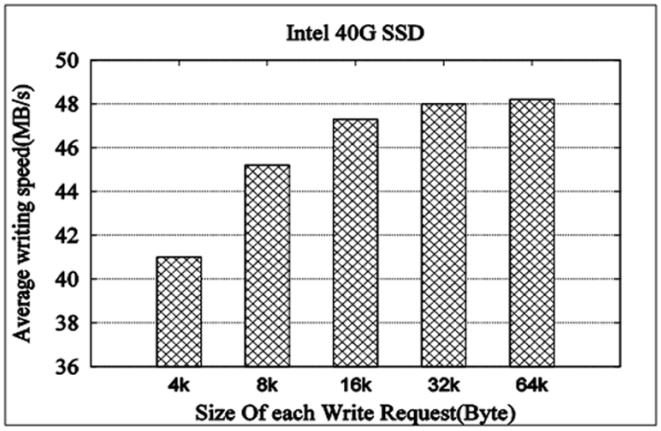


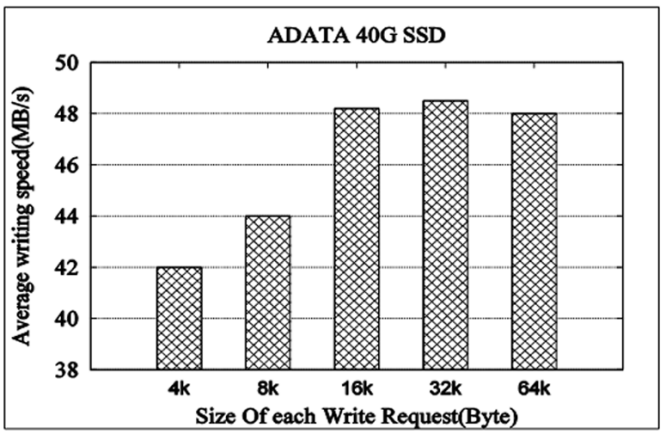
FIGURE 1. System architecture

3.2. **Observations about writing speed of SSDs.** We have employed three SSDs to conduct our experiments: Intel 320 40G SSD, Intel 320 160G SSD, and ADATA S599 40G SSD. Note that the initial capacity utilization of the three SSDs is zero. Two experiments, that is, "changing the size of the write request" and "changing the update ratio of the written data", were performed for each SSD to measure the writing speed of SSDs. The workloads for the experiments included actual workloads by MSR [8] and simulated workloads by Iometer [9]. The written data can be separated into the append data (A_data) and the update data (U_data), where A_data can affect SSDs' space utilization.

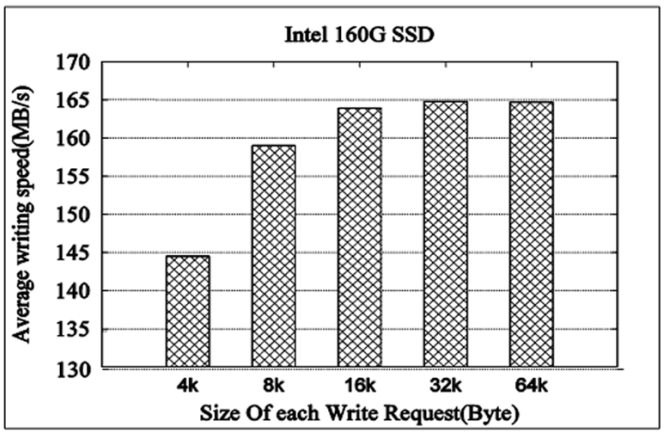
In the first experiment, we observe the relationship between the size of the write request and the writing speed of SSDs. In Figure 2(a) and Figure 2(b), we used two SSDs: Intel 320 40G SSD and ADATA S599 40G SSD, to conduct experiments. Total written data were 10GB, where A_data and U_data in the workloads were 5GB, respectively. The size of each write request was among 4KB, 8KB, 16KB, 32KB, and 64KB. We can observe



(a) Writing speed for Intel 40G SSD



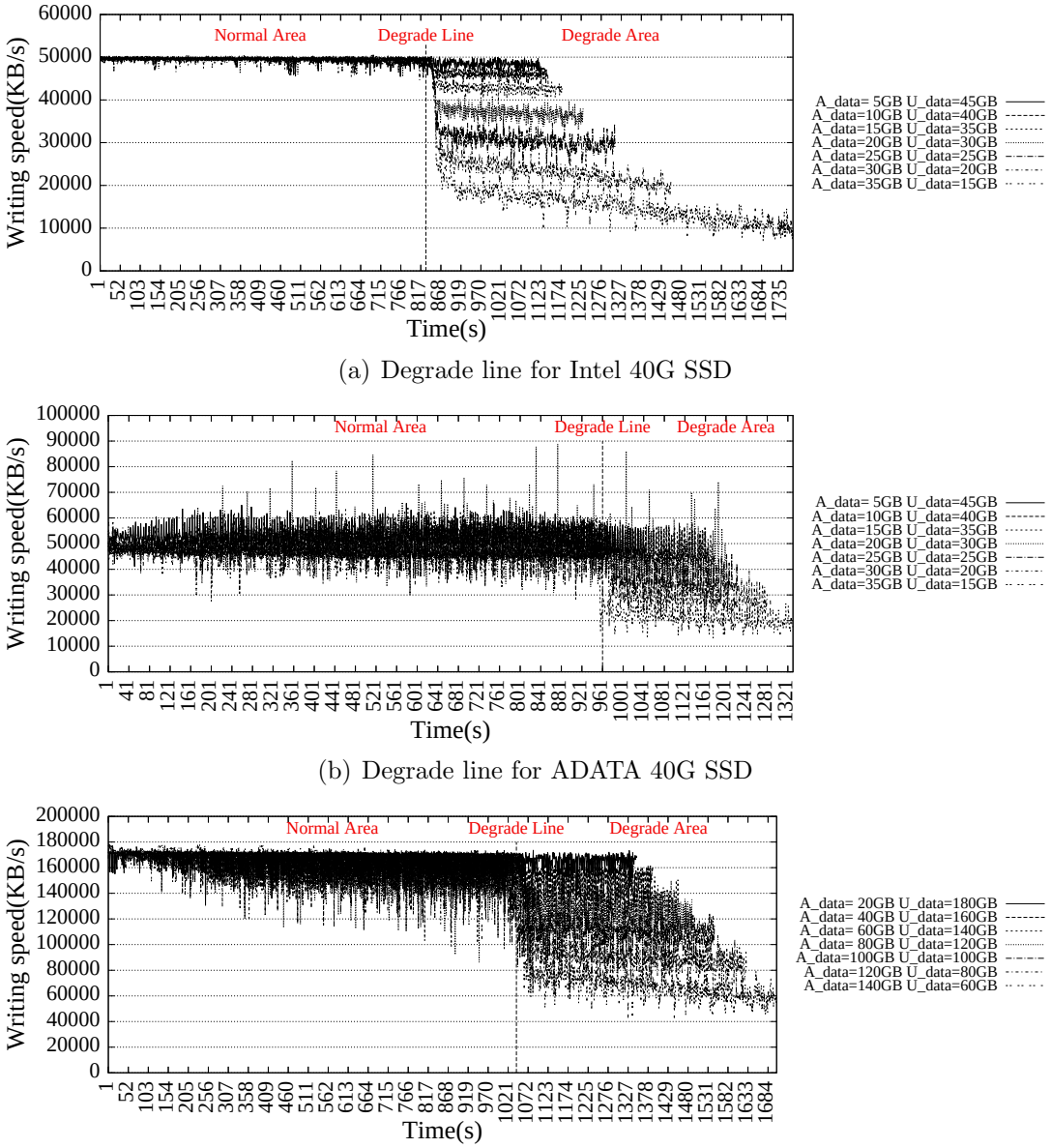
(b) Writing speed for ADATA 40G SSD



(c) Writing speed for Intel 160G SSD

FIGURE 2. Writing speed of SSDs

that large write requests can have better writing speed than that of small write requests. This is because large write requests can efficiently utilize the total bandwidth of SSDs. As shown in Figure 2(c), we also used the same workloads for Intel 320 160G SSD and got the similar results. Overall, the size of the write request can be an important factor for SSDs and large write requests could tend towards high writing speed for SSDs. When a write request is large enough (e.g., $>$ 32KB), its writing speed will become stable for all SSDs used in the experiments. Because the initial capacity utilization of the three SSDs is zero, we use the stable writing speed as the initial average writing speed.



(a) Degrade line for Intel 40G SSD

(b) Degrade line for ADATA 40G SSD

(c) Degrade line for Intel 160G SSD

FIGURE 3. Degrade line of SSDs

In the second experiment, we observe the relationship between the update ratio of written data and the writing speed of SSDs. Total written data were 50GB for Intel 320 40G SSD and ADATA S599 40G SSD, and 200GB for Intel 320 160G SSD. The size of the write request was 64KB and the update ratio of written data was according to the size of U_data. As shown in Figure 3(a), Figure 3(b), and Figure 3(c), we can identify the normal area and the degrade area by a degrade line. The normal area denotes that the writing speed will not be affected significantly and the degrade area denotes that the writing speed will degrade gradually due to garbage collection inside SSDs. The degrade line is to separate the normal area from the degrade area. The reason for the degrade area is that when a lot of data were appended to SSDs, available free space will become low and the activities of garbage collection will be triggered to reclaim free space. Furthermore, we can observe that when the writing speed is in the degrade area, the higher ratio of append data can consume more free space and cause more activities of garbage collection such that the writing speed will degrade significantly. We also summarize the declining speed ratios in Table 1 which contains 7 declining regions for different ratios of append data.

Table 1. Declining speed ratios of SSDs

Region Number	Ratio of Append Data	Intel 40G	Intel 160G	ADATA 40G
0	12.5%	98%	100%	99.5%
1	25%	92.7%	93.7%	94.6%
2	37.5%	85.7%	81.2%	91.9%
3	50%	73.4%	75%	85.9%
4	62.5%	59.7%	68.7%	72.1%
5	75%	40.3%	55%	56.8%
6	87.5%	20.6%	41.8%	42.4%

Table 2. Parameters used in the distribution rules

Average writing speed in the normal area for SSD storage j	W_j
Declining speed ratio of region number s for SSD storage j	α_j^s
Space utilization ratio of region s for SSD storage j	U_j^s

An SSD's current writing speed is the average writing speed multiplied by the declining speed ratio. Therefore, a small declining speed ratio means that the SSD's current writing speed has been reduced. Therefore, different SSDs exhibit different writing speeds, and different space utilization of the same SSD also exhibit different declining speed ratios. Therefore, how to distribute the append data among SSDs for slowing the declining speed ratio will become an important factor for SSDs.

Based on the experimental results, the distribution rules must consider (1) the writing speed of SSDs, (2) the size of the write request, (3) the update ratio of the written data, and (4) the declining speed ratio of SSDs.

3.3. **Consideration 1: A distribution rule for append data.** In the distribution rule for append data, we consider the writing speed of SSDs, the size of the write request, and the declining speed ratio of SSDs. Therefore, for each SSD storage j, we can calculate its $Budget_j$ whose formula is the following: Note that the parameters used in the distribution rules are listed in Table 2. For example, as shown in Table 1, the declining speed ratio of region $s = 2$ for Intel 320 40G SSD (i.e., $\alpha^2_{Intel40G}$) is 85.7%.

$$Budget_j = \frac{W_j \times \left(\alpha_j^s \times \left(1 - U_j^s\right)\right)}{\sum_{k=0}^{n} W_k \times (\alpha_k^{s_k} \times (1 - U_k^{s_k}))} \tag{1}$$

$Budget_j$ denotes the unit weight of the current writing speed ratio for SSD storage j under region s. The current writing speed of SSD storage j with small $Budget_j$ can be better than that with large $Budget_j$. Large $Budget_j$ means that the current writing speed of SSD storage j has been decreased. In particular, $\left(1 - U_j^s\right)$ denotes the unused space ratio for SSD storage j in region s and $\alpha_j^s \times \left(1 - U_j^s\right)$ denotes the declining speed ratio of the unused space ratio for SSD storage j in region s. $W_j \times \left(\alpha_j^s \times \left(1 - U_j^s\right)\right)$ denotes the current writing speed that has dropped to α_j^s of W_j for SSD storage j in region s. $\sum_{k=0}^{n} W_k \times (\alpha_k^{s_k} \times (1 - U_k^{s_k}))$ are the sum of the current writing speed for all storages under their current region s_k. The design idea of $Budget_j$ in the distribution rule is that we can fairly distribute append data to SSDs and smoothly decrease $Budget_j$ to help with a moderate decline in overall speed for SSDs.

Based on the previous experimental results, we can set a threshold (e.g., 32KB) for the size of the write request. When the size of the write request is larger than the threshold, the distribution rule will forward the write request to the current fastest storage j whose

$Budget_j$ is greater than 0. When the size of the write request is less than the threshold, the distribution rule will forward the write request to SSDs in a sequential order. This is because small write requests will not totally utilize SSDs' bandwidth and data can be fairly written to each SSD storage.

3.4. **Consideration 2: A distribution rule for update data.** In the distribution rule for update data, we consider the writing speed of SSDs, the update ratio of the written data (i.e., hot/cold data), and the declining speed ratio of SSDs.

The design idea of $Budget_j$ in the distribution rule is that we can fairly distribute update data to SSDs and smoothly decrease $Budget_j$ to help with a moderate decline in overall speed for SSDs. Therefore, the distribution rule should avoid large $Budget_j$. Assume that SSD storage a has the maximum $Budget_a$ and SSD storage b has the minimum $Budget_b$. When the write request is update data, the distribution rule will use the hot/cold data detection method to determine whether the write request is hot or cold data. Usually, hot data could cause more invalid data and cold data could occupy more free space. If the write request is hot data and will enter into SSD storage b, the distribution rule will forward the write request to SSD storage a. On the other hand, when cold data will enter into SSD storage a, the distribution rule will forward the cold data to SSD storage b. If the write request is cold data, it means that the write request could occupy more free space because the cold data could not be deleted or updated frequently. If the write requests (that contain a lot of cold data) are written to an SSD, the SSD's available free space will become low and the activities of garbage collection will be triggered to reclaim free space such that its *Budget* could become large. The situation is like a lot of data are appended to SSDs. Therefore, we treat hot and cold data differently in the distribution rule. Therefore, the distribution rules for append data and update data can help with a moderate decline in overall speed for SSDs by the $Budget_j$ formula.

4. **Conclusions.** In the paper, we analyze and consider the performance by generating a high volume of writes/updates on SSDs. We provide a detailed analysis on data distribution for SSD-based storage systems to achieve a moderate decline in overall speed for SSD-based storage systems. We propose two distribution rules for append data and update data by considering the observations: (1) the size of the write request can be an important factor for SSDs and large write requests can tend towards the high writing speed for SSDs; (2) the higher ratio of append data can consume more free space and cause more activities of garbage collection such that the writing speed will degrade significantly. The proposed distribution rules must consider the information of append/update data and storage devices, and effectively distribute data with a moderate decline in overall speed for SSDs.

For future research, we should further explore different workloads and the designs of SSD-based storage systems. Especially, how to efficiently run I/O-intensive applications (i.e., sorting) on virtual-machine hybrid storage systems will become an important research topic. Furthermore, a sophisticated customization of SSD-based storage systems and tool platform will become important issues.

Acknowledgement. This work was partially supported in part by a research grant from the Ministry of Science and Technology under Grant MOST 103-2221-E-011-065-MY2.

REFERENCES

[1] C.-K. Kang, Y.-J. Cai, C.-H. Wu and P.-C. Hsiu, A hybrid storage access framework for high-performance virtual machines, *ACM Trans. Embedded Computing Systems*, vol.13, 2014.

[2] K. Liu, X. Zhang, K. Davis and S. Jiang, Synergistic coupling of SSD and hard disk for QoS-aware virtual memory, *IEEE International Symposium on Performance Analysis of Systems and Software*, pp.24-33, 2013.

[3] D. Jiang, Y. Che, J. Xiong and X. Ma, uCache: A utility-aware multilevel SSD cache management policy, *IEEE International Conference on High Performance Computing and Communications*, pp.391-398, 2013.
[4] R.-S. Liu, C.-L. Yang, C.-H. Li and G.-Y. Chen, DuraCache: A durable SSD cache using MLC NAND flash, *ACM/IEEE Design Automation Conference*, pp.1-6, 2013.
[5] Y. Zhu, Y. Yu, W. Y. Wang, S. S. Tan and T. C. Low, A balanced allocation strategy for file assignment in parallel I/O systems, *IEEE Networking, Architecture and Storage*, pp.257-266, 2010.
[6] S. Jung, Y. Lee and Y. H. Song, A process-aware hot/cold identification scheme for flash memory storage systems, *IEEE Consumer Electronics*, vol.56, no.2, pp.339-347, 2010.
[7] D. Park and D. H. C. Du, Hot data identification for flash-based storage systems using multiple bloom filters, *IEEE Mass Storage Systems and Technologies*, pp.1-11, 2011.
[8] Microsoft, *SNIA IOTTA Repository: MSR Cambridge Block I/O Traces*, http://iotta.snia.org/traces/list/BlockIO.
[9] *Iometer Website*, http://www.iometer.org/.