IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON SEQUENTIAL DATA MODELING AND ITS EMERGING APPLICATIONS

Received September 20, 2017, accepted October 16, 2017, date of publication October 26, 2017,
date of current version November 14, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2766667

Efficient Sequential Data Migration Scheme Considering Dying Data for HDD/SSD Hybrid Storage Systems

MINGWEI LIN[1], RIQING CHEN[2], JINBO XIONG[1], XUAN LI[1], AND ZHIQIANG YAO[1]
[1]College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350117, China
[2]Institute of Big Data for Agriculture and Forestry, Fujian Agriculture and Forestry University, Fuzhou 350002, China

Corresponding author: Riqing Chen (riqing.chen@fafu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61502102 and Grant 61402109, in part by the Natural Science Foundation of Fujian province, China, under Grant 2016J05149 and Grant 2015J05120, and in part by the Distinguished Young Scientific Research Talent Training Plan in Universities of Fujian province (2017, 2015).

ABSTRACT Because the solid-state drive (SSD) shows high access performance, it is usually integrated into existing hard disk drive (HDD)-based storage hierarchy to form HDD/SSD hybrid storage systems. To further improve the performance of HDD/SSD hybrid storage systems, data migration schemes have been put forward to migrate the sequential data between HDD and SSD. However, the existing schemes cannot be aware of dirty data in the buffer and then incur a large number of unnecessary page migrations. In this paper, we devise an efficient data migration scheme considering dying data for HDD/SSD hybrid storage systems. In this scheme, a new liveness state, called the dying state, is utilized to identify the live data, which will become dead shortly due to its corresponding dirty version in the buffer. To decrease the page migration count, this scheme uses the benefit-to-cost ratio to select a block for data migration and copies the up-to-date version of dying data into the free space instead of migrating the dying data. Our experimental results show that our proposed scheme can perform better than existing data migration schemes that are not aware of dying data under various benchmarks.

INDEX TERMS Data migration, dying data, hard disk, hybrid storage systems, solid-state drive.

I. INTRODUCTION

Because hard disk drive (HDD) shows non-volatility, high storage capacity, and low cost, it has been the dominant data storage device in computer systems for decades. However, because of the mechanical components of HDD, the I/O access performance of the HDD-based storage systems could not be improved further and the performance gap between the storage system and CPU is becoming much greater [1]–[3].

NAND flash memory is also a kind of non-volatile storage media, which can retain data even when it is powered off. It has several attractive advantages, such as low power consumption, strong shock resistance, small size, and fast access speed, so it has been widely utilized as the secondary storage devices for consumer electronics. For example, mobile phones and tablet computers are usually equipped with NAND flash memory as their data storage devices. With the quick development of the semi-conductor technology, the storage capacity of a single NAND flash memory chip has been increasing dramatically, while its cost-per-bit has been decreasing. Hence, NAND flash memory based solid-state drives (SSD) have emerged [4]–[6].

Since SSD inherits the outstanding advantages, it is usually used to integrate into existing HDD-based storage hierarchy to improve the access performance of storage systems. Although SSD and HDD share the same logical and physical interfaces. However, SSD has very different physical characteristics from HDD. SSD is based on NAND flash memory, which has many unique characteristics. SSD has three basic I/O operations that are read, write, and erase, respectively. Read operation reads data from a target page, which write operation writes data to a target page. Therefore, their basic access unit is a page. Erase operation erases a block whose valid data have been moved to the free space of NAND flash memory. Hence, its basic access unit is a block [7]. SSD also shows asymmetric costs for basic I/O operations. Its read operation cost is lower than the write operation cost, while

the write operation cost is lower than the erase operation cost. Thirdly, SSD owns an erase-before-write hardware constraint that requires a block to be erased before it is written [8]. Finally, SSD has high read performance, while HDD shows fast write speed [9].

To fully combine their respective advantages and also hide their different physical characteristics, several hybrid storage systems have been put forward to maximize the performance. These hybrid storage systems present both SSD and HDD as a single block device to upper-level system components such as file systems [10]–[13]. To further improve the performance of HDD/SSD hybrid storage systems, they often migrate hot data from HDD to SSD and cold data from SSD to HDD. However, existing data migration schemes simply classify all the data in the HDD/SSD hybrid storage systems into two states: live and dead. They focus on the correct identification of hot and cold data in the live data and are not aware of a kind of special live data, which have dirty versions in the buffer. This kind of live data will become dead shortly since the buffer will flush them into the HDD/SSD hybrid storage systems periodically so as to achieve the data consistency. Hence, migrating this kind of live data will incur unnecessary page migrations and then influence the I/O performance of HDD/SSD hybrid storage systems.

To address the above mentioned problem, an efficient data migration scheme is put forward for HDD/SSD hybrid storage systems. In this scheme, a new liveness state named the dying state is introduced to reclassify the live data. It is observed that the data generated by applications are temporarily placed in the buffer so as to improve the access performance. If the buffered data has an old version in the hybrid storage system, then it is usually called the dirty data that will be flushed into the hybrid storage system for achieving data consistency. In this case, the live data in the hybrid storage system having a dirty version in the buffer will become dead when the dirty version is flushed. In the data migration scheme, this kind of live data that will be dead soon due to its corresponding dirty version in the buffer is called dying data.

This scheme prefers to choose a block containing no dying data or the least dying data for migrating so as to decrease the unnecessary page migrations. Distinguishing these dying data from the real live data provides a new idea for improving the performance of HDD/SSD hybrid storage systems. To the best of our knowledge, this work is the first study to improve the performance of HDD/SSD hybrid storage systems by utilizing the dying data.

It is simple, but really effective for HDD/SSD hybrid storage systems. A series of experiments have been conducted with the TPC-C and OLTP workloads and the experiment results shows that the proposed data migration scheme is superior to existing HDD/SSD hybrid storage systems in terms of the response time and the number of page migrations.

The rest of this paper is organized as follows. The existing work is briefly reviewed in Section II. Section III describes the proposed data migration scheme. The experimental evaluation is given in Section IV. Section V concludes this paper.

II. RELATED WORK

To shorten the performance gap between the storage system and CPU, SSD is integrated into existing storage hierarchy due to its high read performance. HDD/SSD hybrid storage systems can be categorized into two types: SSD-based buffer for HDD and HDD and SSD as the same level of the memory hierarchy. Existing works concerning HDD/SSD hybrid storage systems will be briefly reviewed in this section.

A. SSD-BASED BUFFER FOR HDD

Zhang *et al.* put forward iTransformer, which utilizes a small SSD to schedule requests for the data on disk. To achieve high disk efficiency, it buffers dirty data before writing them back into the disk and prefetches some data in a batch into the SSD [14]. Kim *et al.* proposed to use flash memory as a non-volatile cache for hybrid storage systems and designed an intelligent pinning policy to improve its I/O performance [15]. Yoon *et al.* used the Fully Associated Sector Translation (FAST) method to manage the hybrid storage system. In this storage system, flash memory is utilized as a buffer space [16]. Bisson *et al.* utilized flash memory as a buffer of hybrid storage systems to increase the I/O performance and decrease the disk power consumption [17]. Hsu *et al.* proposed to employ NAND flash memory as a second level cache for hybrid storage system in order to store the frequently used programs [18]. Ryu *et al.* proposed a data prefetching scheme, which prefetches a piece of data into flash memory to reduce the disk energy consumption, for HDD/SSD hybrid disk drives [19]. Shim *et al.* put forward to employ flash memory to cache the write requests [20]. Lv *et al.* put forward a hotness-aware buffer management called HAT for flash-based hybrid storage systems. It adopts SSD as a data buffer between main memory and HDD [21], [22]. Han *et al.* utilized SSD as a buffer for SSD and put forward a new SSD caching scheme to migrate data blocks from HDD to SSD [23].

B. HDD AND SSD AT THE SAME LEVEL OF THE MEMORY HIERARCHY

Xie *et al.* combined the hard disks and flash disks and then proposed a hybrid disk array for data-intensive applications. In this work, a self-adaptive file reallocation strategy is designed to adapt to dynamically changed file access patterns [24]–[26]. Suk *et al.* proposed a hybrid file system, named HybridFS, for hybrid storage systems, which stores the data blocks of a large regular file in HDD and places the metadata in SSD [27]. Jo *et al.* presented a hybrid copy-on-write storage system for virtual environments, which stores read-only template disk images in SSD and remaps write operations into HDD [28]. Fisher *et al.* put forward a new hybrid storage system, which combines the fast random read speed of SSD with the high sequential access speed and the large data storage capacity of HDD [29]. Bai *et al.* put forward an unified

flash memory and hard disk translation layer, named FDTL, to manage both HDD and SSD [30]. No *et al.* proposed a new form of file system, called N-hybrid, which can support both HDD and SSD [31]. Yang *et al.* put forward a time-sensitive hybrid storage model. It is designed to map hot and read-intensive pages into SSD and cold or write-intensive pages into HDD [32]. Xiao *et al.* put forward a hybrid storage system, called PASS, to tradeoff between the I/O performance and data discrepancy between the SSD and HDD [33]. Hui *et al.* tried to put forward an energy-efficient hybrid storage system, called E-HASH, which consists of a SSD and multiple HDDs. E-HASH adopts the SSD to store the most frequently read data and HDDs to reserve a log of update distance between currently accessed I/O blocks and their corresponding original I/O blocks and handle all the write requests [34]. Chen *et al.* put forward a high performance hybrid storage system, called Hystor, which identifies critical blocks that can incur long latencies or are the file system metadata and stores them into SSD [35]. Kim *et al.* studied a hybrid data storage system called HybridStore, which can help administrators improve the capacity planning and its performance [36]. Lv *et al.* put forward a novel probabilistic data replacement scheme, called HyPro, for flash-based hybrid storage system, which moves the data between HDD and SSD probabilistically based on the dynamically changed data access pattern [37]. Koltsidas *et al.* designed to use the SSD and HDD at the same level of memory hierarchy and map a data page to only one of these two disks according to the workload [38]. The storage architecture places read-intensive pages into the flash disk and maps write-intensive pages to the hard disk. Wu et al. exploited the access pattern for improving the performance of flash and disk hybrid storage system [39].

The data migration, an important part in HDD/SSD hybrid storage systems, has a great effect on the storage performance. Hence, existing hybrid storage systems focus on how to decide the data is hot or cold. Then they move cold data to HDD and map hot data into SSD [32]. However, none of them eliminate unnecessary page migrations by considering the data that have corresponding new version in the buffer.

III. PROPOSED DATA MIGRATION SCHEME

In this section, our proposed data migration scheme will be presented for hybrid storage systems. First, a concept of a new liveness state, called dying state, will be presented to redefine the live data in the hybrid storage systems. Then, the detailed description of our data migration scheme will be given.

A. DYING DATA

In the existing hybrid storage systems, the data are simply classified into two types: live and dead, as presented in Fig. 1. When some data is written to the hybrid storage device for the first time, the state of the data is set to be live. Later if the data is updated and corresponding new data is written to the hybrid storage device, then the original data will become dead. When the live data is deleted from the hybrid storage device, its state also becomes dead.

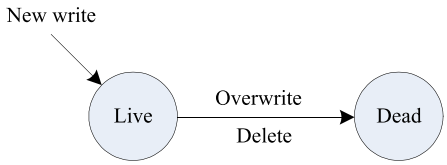

FIGURE 1. Transitions between live and dead states.

To improve the I/O performance for hybrid storage system, it is observed that a buffer is widely utilized to cache a portion of data that may be accessed by user applications recently. When the host reads a file, the buffer stores the data, which are read and/or prefetched from hybrid storage device. When the host writes a file, the new data are temporarily placed in the buffer instead of being written directly into the hybrid storage device. The buffered new data will be flushed to hybrid storage device by the virtual memory manager of Linux operating system for keeping the data consistency [40]–[42]. Our proposed efficient data migration scheme is to distinguish the live data that have the up-to-date version in the buffer from the live data, which do not have corresponding copy in the buffer, and then exploit this buffer information to improve the I/O access performance of hybrid storage system.

In this paper, a new liveness state, called the dying state, is introduced for the data that are stored in hybrid storage device. Therefore, the liveness states of the data depicted in Fig. 1 are further categorized into three types of state, which are the live, dead, and dying, respectively. The dying data are special kind of live data in hybrid storage device, which will become dead shortly when its corresponding up-to-date version in the buffer is flushed into the hybrid storage device. As depicted in Fig. 2, if the copy of the live data in the buffer is modified, its state is changed to dying. If the modified copy is later flushed into the hybrid storage device or the host deletes the dying data, then it becomes dead. If the modified copy of dying data in the buffer is lost because of the sudden power failure of the host, its state becomes live. If the live data is deleted by the host directly, its state is changed to dead.

As depicted in Fig. 2, it can be seen that the dying state is a transient state between the live and dead states. The difference between the dying data and live data is that the dying data will become dead in the near future since its up-to-date copy in

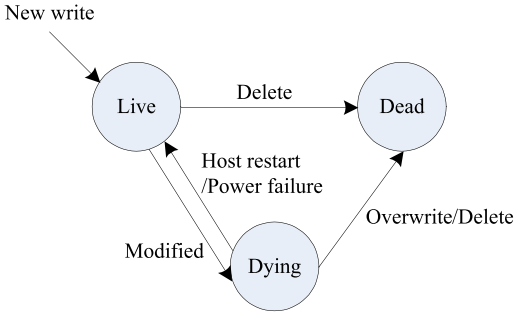

FIGURE 2. Transitions between live, dead, and dying states.

the buffer will be flushed into the hybrid storage device. How soon the up-to-date copy in the buffer will be flushed depends on the buffer management strategy. In the Linux operating system, the time interval is set to 30 seconds by default.

B. DATA MIGRATION SCHEME

In this paper, the hybrid storage system is prototyped as a kernel module in the Linux operating system with the minimal changes. It presents both the HDD and SSD as a single virtual block device and hides the different physical characteristics of them from other system components such as the file system. It divides the entire storage space of HDD into a fixed number of blocks whose size is the same as that of the block in SSD.

The dying data will be dead shortly after its corresponding up-to-date version in the buffer is written back into the hybrid storage device, so migrating the dying data will lead to a large number of unnecessary page migrations. Hence, the designing idea of our proposed data migration scheme is to minimize the number of page migrations by considering the dying data.

In our proposed data migration scheme, the data migration procedure is composed of three steps, which are (1) selecting a victim block to migrate data, (2) migrating data between HDD and SSD, and (3) updating the device mapping table. We try to improve the first two steps by considering the dying data.

1) SELECTING A VICTIM BLOCK

Some of existing hybrid storage systems choose one file for data migration according to the access pattern [24]–[26]. If the selected file is too large, then it will result in a large number of page migrations. The rest ones usually select one page for data migration based on its history access information [32], [38], so the hybrid storage system has to utilize a large amount of main memory to store the history access information of all the pages. Moreover, none of them consider the dying data in the buffer.

In this paper, our proposed data migration scheme sets the granularity of page migration to a block and then computes the benefit-to-cost ratio of migrating a block as follows:

$$\frac{benefit}{cost} = \frac{\bar{h} * \left(D - \min\left(d, \frac{D}{2}\right)\right)}{2\left(N - D\right)} \tag{1}$$

where the term D represents the number of dead pages in the block, the term d represents the number of dying pages in the block, the term $\bar{h}$ represents the normalized hot degree of the block, the term N represents the total number of pages in the block. The term $N - D$ denotes the number of live data, which will be migrated, and the number of dying data whose version in the buffer will be flushed. Since the live data should be read first and then be written into other storage device such as HDD or SSD, $2\left(N - D\right)$ is defined as the cost of data migration. The term $\bar{h} * \left(D - \min\left(d, \frac{D}{2}\right)\right)$ is defined as the benefit.

The hot degree of each block is calculated by utilizing the exponentially weighted moving average method as follows:

$$\begin{aligned} h\left(i\right) &= \beta * \left(t_i - t_{i-1}\right) + \beta * h\left(i-1\right) \\ &= \beta * \left(t_i - t_{i-1}\right) + \beta * \left(1-\beta\right) * \left(t_{i-1} - t_{i-2}\right) \\ &\quad + \beta * \left(1-\beta\right)^2 * \left(t_{i-2} - t_{i-3}\right) \\ &\quad + \cdots + \beta * \left(1-\beta\right)^{i-2} * \left(t_2 - t_1\right) + \left(1-\beta\right)^{i-1} \\ &\quad * h\left(1\right) \end{aligned} \tag{2}$$

with:

$$h\left(1\right) = t_2 - t_1 \tag{3}$$

where $\{t_1, t_2, t_3, \cdots, t_i\}$ are the past update times of this block, β is a coefficient and its value is set to [0,1].

When the data will be migrated from the HDD into SSD, the normalized hot degree of the block is calculated as follows

$$\bar{h} = \frac{1}{h} \tag{4}$$

When the data will be migrated from the SSD into HDD, the normalized hot degree of the block is obtained as $\bar{h} = h$.

Hence, the block with the highest benefit-to-cost ratio will be selected as a victim block for data migration.

2) MIGRATING DATA BETWEEN HDD AND SSD

Our proposed data migration scheme migrates the data in the victim block as shown in Fig. 3.

```
Migrating_data(block b)
For (each page p_i in the b)
{
  if p_i is dead
    does not migrate p_i;
  else if  p_i is dying
    flush its up-to-date version in the buffer into the storage device;
  else
    migrate p_i;
}
```

FIGURE 3. The process for migrating data.

It first checks the status of each page in the victim block. If the page is dead, it will not be migrated between HDD and SSD. If it is dying, its up-to-date version in the buffer will be flushed into the storage device to keep the data consistency. Otherwise, it will be migrated between HDD and SSD.

To help the reader understand the migration process easily, an example is shownd to compare our proposed data migration scheme with existing hybrid storage systems as illustrated in Fig. 4 and Fig. 5.

As shown in Fig. 4, it can be seen that the existing hybrid storage systems do not distinguish dying pages from live ones and generate six page migrations. However, dying pages have corresponding up-to-date data in the buffer. Hence, if they are migrated, they will become dead since their up-to-date data in the buffer will be flushed to the hybrid storage device shortly. In this case, migrating dying pages will result in unnecessary

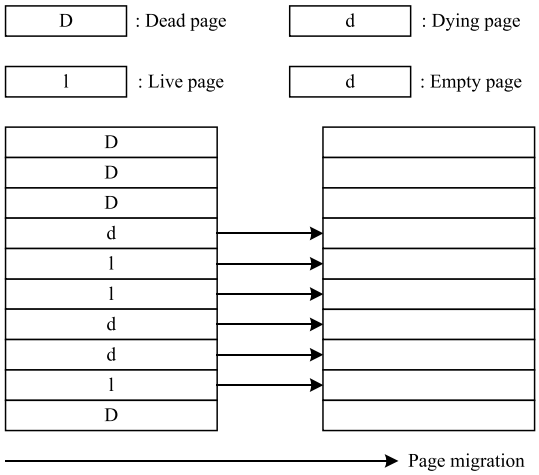

FIGURE 4. The process of page migration for existing HDD/SSD hybrid storage systems.

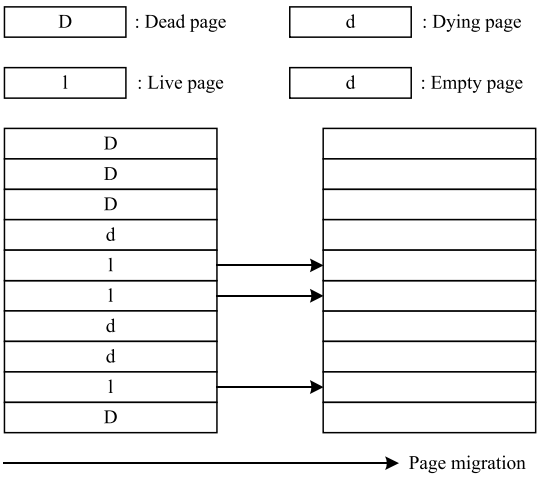

FIGURE 5. The process of page migration in our proposed data migration scheme.

page migrations. Be aware of this information, dying pages in the hybrid storage device are distinguished from live pages by our proposed data migration scheme, which does not migrate the dying pages during the page migration process. As depicted in Fig. 5, it results in only three page migrations and other three unnecessary page migrations are eliminated.

IV. PERFORMANCE EVALUATION

In this section, we first show the experiment setup and then the experimental results for various HDD/SSD hybrid storage systems are presented.

A. EXPERIMENT SETUP

We implemented a prototype system and then conducted a series of experiments to evaluate our proposed data migration scheme. This prototype system consists of a storage manager, which uses the B+-tree to manage the data stored in the hybrid storage device, and a buffer manager, which utilizes the Least Recently Used (LRU) algorithm for managing the data cached in the buffer. This prototype system is developed in C++ and running on a machine, which is equipped with a 2.66GHz Intel processor and 4 GB of main memory. We use Ubuntu 16.04 LTS with the Linux kernel 4.40 and Ext3 file system with the default configurations. In order to minimize the interference, Ubuntu operating system and its home directory are stored in a dependent hard disk. Another two disks, which are a flash disk and a hard disk, are used to place the user data. The flash disk is a 32 GB Intel X25-E SSD, while the hard disk is a Seagate Cheetah 15K.5 HDD. TABLE I lists the specifications of these two disks in detailed. The HDD is connected to the host by using the SAS interface and the SSD uses the SATA interface.

TABLE 1. Specifications of HDD and SSD in our experiments.

Feature	SSD	HDD
Capacity	32 GB	73 GB
Interface	SATA	SAS
Bandwidth	250 MB/s (read), 170 MB/s (write)	125 MB/s

Our proposed data migration scheme is evaluated by using two workloads, which are the TPC-C and OLTP, respectively. The detailed specifications of these two workloads are listed in TABLE II. The hybrid storage systems in our experiments are Koltsidas's work [38], Yang's work [32], our proposed data migration scheme. The coefficient β is set to 0.5.

TABLE 2. Specifications of TPC-C and OLTP.

Type	Read count	Write count	Total count
TPC-C	135114 (77.1%)	400738 (22.9%)	1751882
OLTP	470678 (77.5%)	136713 (22.5%)	607391

B. EXPERIMENTAL RESULTS

Fig. 6 and Fig. 7 depict the response times of three hybrid storage systems under TPC-C and OLTP benchmark. It can be seen that our work consumes less response time than other two hybrid storage systems. That is because our work employs the exponentially weighted moving average (EWMA) method for calculating the hot degree of each

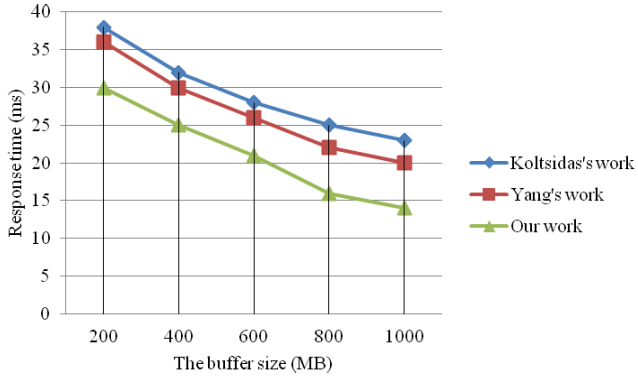

FIGURE 6. The response times of three hybrid storage systems for TPC-C.

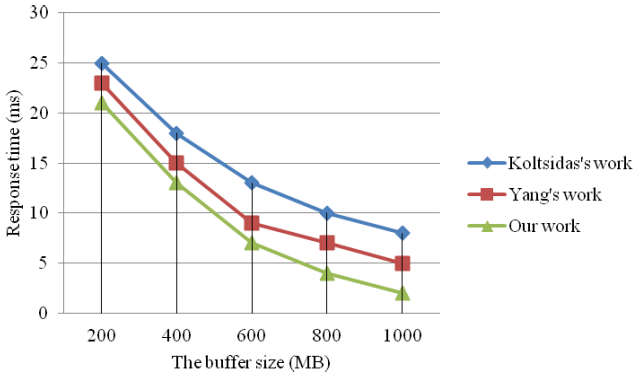

FIGURE 7. The response times of three hybrid storage systems for OLTP.

block and then it can predict the block accurately, which will be accessed in the near future.

Compared with Yang's work, our work reduces the response time by at least 16.7% under the TPC-C benchmark and 8.7% under the OLTP benchmark.

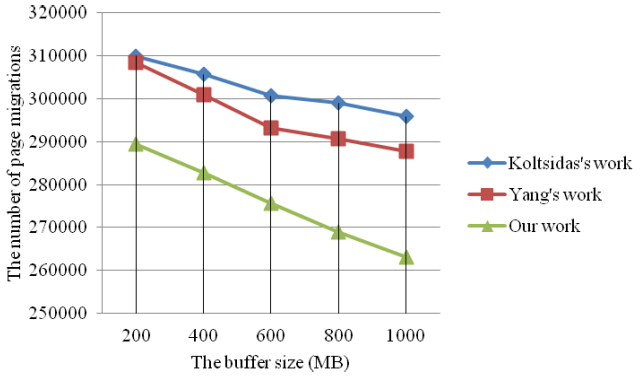

FIGURE 8. The number of page migrations of various hybrid storage systems for TPC-C.

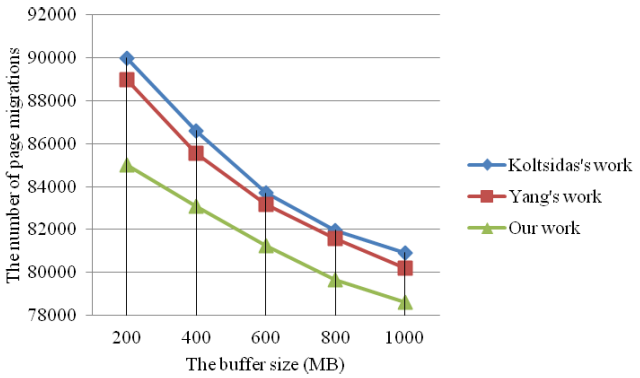

FIGURE 9. The number of page migarations of various hybrid storage systems for OLTP.

Fig. 8 and Fig. 9 illustrate the number of page migrations of various hybrid storage systems under the TPC-C and OLTP benchmark. It can be seen that our work can result in the least number of page migrations. That is because our work is aware of distinguishing the dying data from the live data and do not migrate the dying data. Hence, a large number of unnecessary page migration are eliminated in our work.

Compared with Yang's work, our work reduces the number of page migrations by up to 8.6% under the TPC-C benchmark and 4.4% under the OLTP benchmark.

V. CONCLUSIONS

To reduce unnecessary page migrations for hybrid storage systems, we put forward an efficient data migration scheme in this paper. It categorizes all data in the hybrid storage systems into three types: live data, dead data, and dying data. The live data, which has corresponding up-to-date version in the buffer, is defined as the dying data. The dying data will become dead shortly when its up-to-date version in the buffer is flushed into the hybrid storage device. It exploits the information of dying data to compute the benefit-to-cost ratio of all the blocks in the hybrid storage system and choose the block with the largest benefit-to-cost ratio as a victim for migrating data. During the data migration process, our proposed data migration scheme does not copy the dying data from one device to another device. To achieve high data consistency, the up-to-date version of the dying data is flushed into the storage device. Experiments are conducted with the TPC-C and OLTP benchmarks and results show that our proposed data migration scheme can reduce the response time and the number of page migrations efficiently.

REFERENCES

[1] Y. Xia, Y. Liu, J. Liu, and Q. Zhu, "Modeling and performance evaluation of BPEL processes: A stochastic-Petri-net-based approach," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 42, no. 2, pp. 503–510, Mar. 2012.

[2] Y. Xia, M. Zhou, X. Luo, S. Pang, and Q. Zhu, "A stochastic approach to analysis of energy-aware DVS-enabled cloud datacenters," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 1, pp. 73–83, Jan. 2015.

[3] Y. Xia, M. Zhou, X. Luo, S. Pang, and Q. Zhu, "Stochastic modeling and performance analysis of migration-enabled and error-prone clouds," *IEEE Trans. Ind. Informat.*, vol. 11, no. 2, pp. 495–504, Apr. 2015.

[4] Y. Lu, J. Shu, J. Guo, and P. Zhu, "Supporting system consistency with differential transactions in flash-based SSDs," *IEEE Trans. Comput.*, vol. 65, no. 2, pp. 627–639, Feb. 2016.

[5] C. Matsui, A. Arakawa, C. Sun, and K. Takeuchi, "Write order-based garbage collection scheme for an LBA scrambler integrated SSD," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 2, pp. 510–519, Feb. 2017.

[6] B. Van Houdt, "On the power of asymmetry and memory in flash-based SSD garbage collection," *Perform. Eval.*, vol. 97, pp. 1–15, Mar. 2016.

[7] S. Lee and J. Kim, "Effective lifetime-aware dynamic throttling for NAND flash-based SSDs," *IEEE Trans. Comput.*, vol. 65, no. 4, pp. 1075–1089, Apr. 2016.

[8] J. K. Park, J.-Y. Lee, and S. H. Noh, "Divided disk cache and SSD FTL for improving performance in storage," *J. Semicond. Technol. Sci.*, vol. 17, no. 1, pp. 15–22, 2017.

[9] W.-H. Kang, S.-W. Lee, and B. Moon, "Flash as cache extension for online transactional workloads," *VLDB J.*, vol. 25, no. 5, pp. 673–694, 2016.

[10] Z. Feng, Z. Feng, X. Wang, G. Rao, Y. Wei, and Z. Li, "HDStore: An SSD/HDD hybrid distributed storage scheme for large-scale data," in *Web-Age Information Management* (Lecture Notes in Computer Science), vol. 8597. Heidelberg, Germany: Springer-Verlag, 2014, pp. 209–220.

[11] P. Yang, P. Jin, S. Wan, and L. Yue, "HB-storage: Optimizing SSDs with a HDD write buffer," in *Web-Age Information Management* (Lecture Notes in Computer Science), vol. 7901. Heidelberg, Germany: Springer-Verlag, 2013, pp. 28–39.

[12] Z. Chen, N. Xiao, F. Liu, and Y. Du, "A high performance reliable storage system using HDDs as the backup of SSDs," *J. Comput. Res. Develop.*, vol. 50, no. 1, pp. 80–89, 2013.

[13] P. Yang, P. Jin, and L. Yue, "SH-Sim: A flexible simulation platform for hybrid storage systems," *Int. J. Grid Distrib. Comput.*, vol. 7, no. 3, pp. 61–70, 2014.
[14] X. Zhang, K. Davis, and S. Jiang, "iTransformer: Using SSD to improve disk scheduling for high-performance I/O," in *Proc. IEEE 26th Int. Parallel Distrib. Process. Symp.*, May 2012, pp. 715–726.
[15] Y. J. Kim, S. J. Lee, K. Zhang, and J. Kim, "I/O performance optimization techniques for hybrid hard disk-based mobile consumer devices," *IEEE Trans. Consum. Electron.*, vol. 53, no. 4, pp. 1469–1476, Nov. 2007.
[16] U.-K. Yoon, H.-J. Kim, and J.-Y. Chang, "Intelligent data prefetching for hybrid flash-disk storage using sequential pattern mining technique," in *Proc. 9th IEEE/ACIS Int. Conf. Comput. Inf. Sci.*, Aug. 2010, pp. 280–285.
[17] T. Bisson and S. A. Brandt, "Reducing hybrid disk write latency with flash-backed I/O requests," in *Proc. 15th Int. Symp. Modeling, Anal., Simulation Comput. Telecommun. Syst.*, Oct. 2007, pp. 402–409.
[18] H.-T. Hsu and Y.-W. Bai, "Using NAND flash memory to improve the performance of HDDs," in *Proc. 3rd Can. Conf. Elect. Comput. Eng.*, May 2010, pp. 1–6.
[19] W. Ryu and M. Song, "Design and implementation of a disk energy saving scheme for media players which use hybrid disks," *IEEE Trans. Consum. Electron.*, vol. 56, no. 4, pp. 2382–2386, Nov. 2010.
[20] H. Shim, J.-S. Kim, and S. Maeng, "BEST: Best-effort energy saving techniques for NAND flash-based hybrid storage," *IEEE Trans. Consum. Electron.*, vol. 58, no. 3, pp. 841–848, Aug. 2012.
[21] Y. Lv, B. Cui, X. Chen, and J. Li, "Hotness-aware buffer management for flash-based hybrid storage systems," in *Proc. 22nd ACM Int. Conf. Inf. Knowl. Manage.*, 2013, pp. 1631–1636.
[22] Y. Lv, B. Cui, X. Chen, and J. Li, "HAT: An efficient buffer management method for flash-based hybrid storage systems," *Front. Comput. Sci.*, vol. 8, no. 3, pp. 440–455, 2014.
[23] C. Han, J. Ryu, D. Lee, J. Lee, K. Kang, and H. Shin, "File-system-level flash caching for improving application launch time on logical hybrid disks," in *Proc. IEEE 33rd Int. Perform. Comput. Commun. Conf.*, Dec. 2014, pp. 1–2.
[24] T. Xie and D. Madathil, "SAIL: Self-adaptive file reallocation on hybrid disk arrays," in *High Performance Computing* (Lecture Notes in Computer Science), vol. 5374. Heidelberg, Germany: Springer-Verlag, 2008, pp. 529–540.
[25] T. Xie and Y. Sun, "Dynamic data reallocation in hybrid disk arrays," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 9, pp. 1330–1341, Sep. 2010.
[26] T. Xie and Y. Sun, "PEARL: Performance, energy, and reliability balanced dynamic data redistribution for next generation disk arrays," in *Proc. IEEE Int. Symp. Modeling, Anal. Simulation Comput. Telecommun. Syst. (MASCOTS)*, Sep. 2008, pp. 1–8.
[27] J. Suk and J. No, "Performance analysis of NAND flash-based SSD for designing a hybrid filesystem," in *Proc. 11th IEEE Int. Conf. High Perform. Comput. Commun.*, Jun. 2009, pp. 539–544.
[28] H. Jo, Y. Kwon, H. Kim, E. Seo, J. Lee, and S. Maeng, "SSD-HDD-hybrid virtual disk in consolidated environments," in *Proc. Eur. Conf. Parallel Process.*, vol. 6043. Heidelberg, Germany: Springer-Verlag, 2010, pp. 375–384.
[29] N. Fisher, Z. He, and M. McCarthy, "A hybrid filesystem for hard disk drives in tandem with flash memory," *Computing*, vol. 94, no. 1, pp. 21–68, 2012.
[30] S. Bai, J. Yin, G. Tan, Y.-P. Wang, and S.-M. Hu, "FDTL: A unified flash memory and hard disk translation layer," *IEEE Trans. Consum. Electron.*, vol. 57, no. 4, pp. 1719–1727, Nov. 2011.
[31] J. No, "NAND flash memory-based hybrid file system for high I/O performance," *J. Parallel Distrib. Comput.*, vol. 72, no. 12, pp. 1680–1695, 2012.
[32] P.-Y. Yang, P.-Q. Jin, and L.-H. Yue, "A time-sensitive and efficient hybrid storage model involving SSD and HDD," *Chin. J. Comput.*, vol. 35, no. 11, pp. 2294–2305, 2012.
[33] W. Xiao, X. Lei, R. Li, N. Park, and D. J. Lilja, "PASS: A hybrid storage system for performance-synchronization tradeoffs using SSDs," in *Proc. 10th IEEE Int. Symp. Parallel Distrib. Process. Appl.*, Jul. 2012, pp. 403–410.
[34] J. Hui, X. Ge, X. Huang, Y. Liu, and Q. Ran, "E-HASH: An energy-efficient hybrid storage system composed of one SSD and multiple HDDs," in *Advances in Swarm Intelligence* (Lecture Notes in Computer Science), vol. 7332. Heidelberg, Germany: Springer-Verlag, 2012, pp. 527–534.
[35] F. Chen, D. A. Koufaty, and X. Zhang, "Hystor: Making the best use of solid state drives in high performance storage systems," in *Proc. Int. Conf. Supercomput.*, 2011, pp. 22–32.
[36] Y. Kim, A. Gupta, B. Urgaonkar, P. Berman, and A. Sivasubramaniam, "HybridStore: A cost-efficient, high-performance storage system combining SSDs and HDDs," in *Proc. 19th Annu. IEEE/ACM Int. Symp. Modeling, Anal., Simulation Comput. Telecommun. Syst.*, Jul. 2011, pp. 227–236.
[37] Y. Lv, X. Chen, G. Sun, and B. Cui, "A probabilistic data replacement strategy for flash-based hybrid storage system," in *Web Technologies and Applications* (Lectors Notes in Computer Science), vol. 7808. Heidelberg, Germany: Springer-Verlag, 2013, pp. 360–371.
[38] I. Koltsidas and S. D. Viglas, "Flashing up the storage layer," *Proc. VLDB Endowment*, vol. 1, no. 1, pp. 514–525, 2008.
[39] X. Wu and A. L. N. Reddy, "Managing storage space in a flash and disk hybrid storage system," in *Proc. IEEE Int. Symp. Modeling, Anal. Simulation Comput. Telecommun. Syst.*, Sep. 2009, pp. 610–613.
[40] M. Lin, S. Chen, G. Wang, and T. Wu, "HDC: An adaptive buffer replacement algorithm for NAND flash memory-based databases," *Optik-Int. J. Light Electron Opt.*, vol. 125, no. 3, pp. 1167–1176, 2014.
[41] M. Lin, Z. Yao, and T. Huang, "F-LRU: An efficient buffer replacement algorithm for NAND flash-based databases," *Optik-Int. J. Light Electron Opt.*, vol. 127, no. 2, pp. 663–667, 2016.
[42] X.-L. Liao and S.-M. Hu, "Bridging the information gap between buffer and flash translation layer for flash memory," *IEEE Trans. Consum. Electron.*, vol. 57, no. 4, pp. 1765–1773, Nov. 2011.

MINGWEI LIN received the B.S. degree in software engineering and Ph.D. degree in computer science and technology from Chongqing University, Chongqing, China, in 2009 and 2014, respectively. From 2015 to 2016, he was a Lecturer with the Faculty of Software, Fujian Normal University, Fuzhou, China. Since 2017, he has been an Associate Professor with the College of Mathematics and Informatics, Fujian Normal University. He has authored or co-authored over 20 research papers as the lead author in international journals and conference proceedings. His research interests include storage systems and embedded systems. He was the recipient of the CSC-IBM Chinese Excellent Student Scholarship in 2012.

RIQING CHEN received the B.Eng. degree in communication engineering from Tongji University, Shanghai, China, in 2001, the M.Sc. degree in communications and signal processing from Imperial College London, London, U.K., in 2004, and the Ph.D. degree in engineering science from the University of Oxford, Oxford, U.K., in 2010. Since 2014, he has been a Full Professor with the Faculty of Computer and Information Sciences, Fujian Agriculture and Forestry University, Fuzhou, China. His research interests include big data and visualization, cloud computing, consumer electronics, flash memory, and wireless sensor networking.

JINBO XIONG received the B.S. degree in electronics and information engineering from the Hebei University of Engineering, Handan, China, 2003, the M.S. degree in communication and information systems from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2006, and the Ph.D. degree in computer system architecture from Xidian University, Xi'an, China, in 2013. Since 2006, he has been with Fujian Normal University, Fuzhou, China, where he is currently an Associate Professor. His research interests include NAND flash memory, mobile data security, and big data security.

XUAN LI received the B.S. degree in mathematics and applied mathematics and Ph.D. degree in computer science and technology from the South China University of Technology, Guangzhou, China, in 2007 and 2012, respectively. From 2012 to 2014, she was a Lecturer with the Information Science School, Guangdong University of Finance and Economics, Guangzhou. Since 2015, she has been an Associate Professor with Fujian Normal University, Fuzhou, China. Her research interests include consumer electronics and computer security.

ZHIQIANG YAO received the B.S. degree in mathematics from Fujian Normal University, Fuzhou, China, in 1989, the M.S. degree in mathematics from East China Normal University, Shanghai, China, in 1992, and the Ph.D. degree in computer system architecture from Xidian University, Xi'an, China, in 2014. Since 1992, he has been with Fujian Normal University, where he is currently a Professor. His current research interests include multimedia security and NAND flash memory.