IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 27, 2017, accepted February 1, 2018, date of publication February 7, 2018, date of current version March 28, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2803302

Hybrid Storage Systems: A Survey of Architectures and Algorithms

JUNPENG NIU[1], (Student Member, IEEE), JUN XU[2], (Senior Member, IEEE), AND LIHUA XIE[1], (Fellow, IEEE)

[1]School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798
[2]HGST Western Digital Corporation, Singapore 417939

Corresponding author: Jun Xu (xujun@ieee.org)

ABSTRACT Data center storage architectures face rapidly increasing demands for data volume and quality of service requirements today. Hybrid storage systems have turned out to be the one of the most popular choices in fulfilling these demands. A mixture of various types of storage devices and structures enables architects to address performance and capacity concerns of users within one storage infrastructure. In this paper, we present an extensive literature review on the state-of-the-art research for hybrid storage systems. First, different types of hybrid storage architectures are explored and categorized thoroughly. Second, the corresponding algorithms and policies, such as caching, scheduling, resource allocation and so on, are discussed profoundly. Finally, the advantages and disadvantages of these hybrid storage architectures are compared and analyzed intensively, in terms of system performance, solid state drive lifespan, energy consumption, and so on, in order to motivate some future research directions.

INDEX TERMS Caching algorithms, data migration, hot data identification, hybrid storage system.

I. INTRODUCTION

In the past decades, the speed of data growing is in exponential. The IDC report [1] estimates that the digital universe doubles in size every two years. In view of its estimation, by 2020, the data may increase to 40 zettabytes. Such tremendous and assorted data requirements create challenging issues for the design of storage infrastructures. Other than the requirement of storage capacity, the data accessing speed and reliability are also key factors influencing the user experience.

Due to the mechanical limitations of Hard Disk Drive (HDD), the traditional storage architecture turns out to be the major performance bottleneck in data intensive systems. With the quick evolution of Flash memory technologies [2], Solid State Drive (SSD) is going to replace HDD in some data centers (DCs) as the IOPS of SSD can be hundreds or even thousands times faster than that of HDD, and throughput can be also times over HDD. Other than that, SSD also brings down energy utilization and provides stable execution under vibration. However, due to the high cost per GB, limited write cycles, and reliability issues of SSD [3], HDD still plays an important role in modern storage systems.

In order to provide both large storage capacity and fast access speed with the minimal cost, it is vital to fully utilize the high capacity of low tier devices with lower accessing speed and lower cost per GB, e.g. HDD, and the high performance of high tier devices with lower capacity and higher cost per GB, e.g. SSD. Hybrid storage architecture is one of the solutions that can provide comparable or close system performance with all fast devices, whereas giving nearly same storage capacity with all lower tier devices. It usually has the ability to automatically promote or demote different types of data across different tiers of storage devices. The data movement is managed via either the host or the drive itself according to the performance and capacity requirements. The hybrid storage system is the main-stream in modern data centers, and will continue to play an important role in the storage infrastructure if the performance and cost difference of storage devices still exists.

In early days, high-end HDDs (such as 15k RPM and 10k RPM) acted as the performance tier, and low-end HDDs (such as 7200 RPM and 5400 RPM) acted as the capacity tier [4]. Nowadays, NAND Flash [2] SSDs are employed to replace high-end HDDs. The next generation of hybrid storage systems have incorporated with the emerging Non-Volatile Memory (NVM) technologies, such as Phase Change Memory (PCM) [5], Spin-Transfer Torque Magnetic random-access memory (STTMRAM) [6], Resistive RAM (RRAM) [7], etc. Table 1 lists the performance and price of

TABLE 1. The performance, power consumption, price and endurance of some non-volatile memories and HDD.

	STTMRAM	PCMS 3D Xpoint	RRAM	Flash NAND	HDD
Read latency	$10-20ns$	$50-100ns$	$250ns$	$10^5 ns$	$3ms$
Power consumption	Medium	Medium	Medium	Very high	Highest
Price (2016)	$200-3000/Gb$	$\leq 0.5/Gb$	$100/Gb$	$\leq 0.05/Gb$	$\leq 0.002/Gb$
Endurance (Nb cycles)	10^{12}	10^8	10^6	10^5	N.A

TABLE 2. The literature of hybrid storage systems (HDDs + SSDs).

Categories	Literature
SSD cache	[26] [27] [28] [29] [30] [31] [32] [33] [34] [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [19] [18] [20] [50] [51] [52] [53] [54] [17] [55] [56] [57] [58] [14] [15] [16] [59] [60] [61] [62] [63] [64] [65]
SSD tier	[66] [67] [68] [69] [70] [71] [72] [73] [74] [75] [76] [77] [78] [79] [80] [81] [82] [83] [84] [79] [85] [86] [87] [88]
SSD as mixed cache and tier	[89] [90] [91] [92] [93] [94]
HDD as the write cache of SSD	[95] [96] [97] [98] [99]

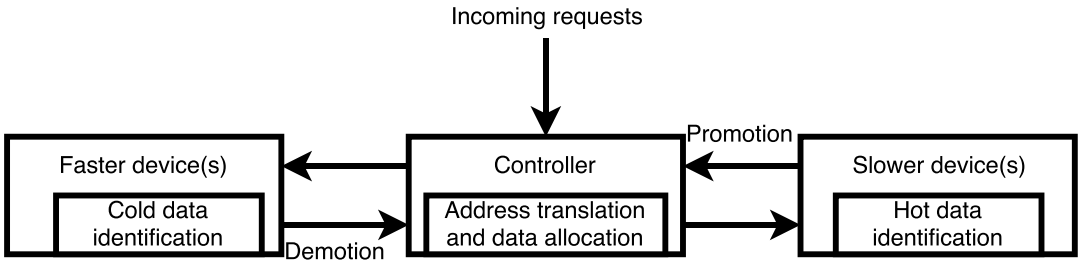


FIGURE 1. General algorithms for hybrid storage system.

some emerged NVMs and HDD. As the performance cost ratio, which is the ratio of performance, the combination of random access speed (IOPS) and sequential access speed (throughput) for various system requirements over the price per GB, of NVMs is increasing, these NVMs with fast speed can be used as the performance tier [8], [9] or cache [10]–[13] in modern hybrid storage systems.

Nowadays, although there are hybrid drive products that consist of HDD and NAND flash, data centers prefer to design their own hybrid structures in the system level, where SSD is still the first choice as the performance tier, HDD and the high capacity Shingled Magnetic Recording (SMR) drives are often utilized as the backup tier [14]–[17]. Many industry players [18]–[25] integrate the SSD devices into the traditional storage system as the faster tier or data cache to improve the system performance with slightly increased cost. Meanwhile, lots of researchers, as listed in Table 2, have also proposed various structures and algorithms to improve one or few aspects of hybrid storage systems. In this paper, we concentrate on the hybrid storage structures that combine HDDs and SSDs, despite the fact that the general idea can be applied to a wide range of hybrid storage systems.

A general hybrid storage architecture is shown in Fig. 1, where different algorithms and policies are used to manage the external and internal data for the tier and cache storage architectures. There are mainly two types of hybrid storage architectures, tiered storage architecture (tiering method) and cache storage architecture (caching method). The fundamental distinction between the two architectures are as follows: 1) Tiering method moves data to the fast storage area as opposed to copying the data in caching method. 2) The time duration that the hot data stay in the faster device is normally longer with tiering method compared with caching method. However, their performances are chiefly influenced by the following four aspects.

- Firstly, data allocation policies are essential to control the data flows among various devices. The data allocation in tiering method allocates the data based on the hotness of the data. Meanwhile, the data allocation in caching method distributes the data to different devices in light of the caching policy utilized, such as read-only, write-back, etc.
- Secondly, address translation mechanism is needed in a hybrid storage system to keep the different locations in different devices for the same data. The design of address translation is important to the speed of data retrieval and the size of memory utilization to store the translation information.
- Thirdly, due to the size limitation of SSD cache/tier compared with main storage HDDs, the SSD cache/tier is

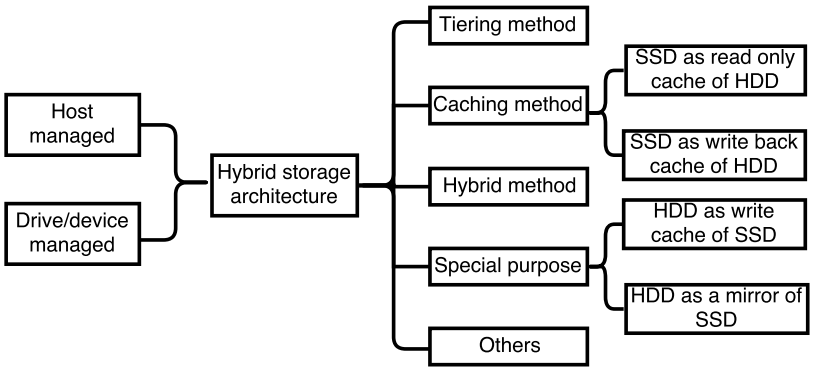


FIGURE 2. The overall categories of the hybrid storage architectures.

better to only store the frequently and recently accessed data, i.e., hot data. Therefore, it is important to have a suitable hot data identification method to identify whether the coming data are hot or not. When the hot data are detected, the data need to be promoted when necessary. A data migration policy is needed to control the hot/cold data flows to improve the future access efficiency. The hot data identification and data migration consider the trade-off between the cost of data migration and the benefit of accessing speed of the faster device, which affects the overall system performance.

- Lastly, a cache scheduling algorithm is necessary for controlling the queuing behaviors, such as the queue size, synchronization, and execution sequence. The designed scheduling algorithm affects the usage efficiency of the storage devices and the cache inside.

In this paper, the hybrid structures are properly categorized and the corresponding algorithms are intensively discussed. In addition, the in-depth comparison points out the advantage and disadvantages of those structures and algorithms under different scenarios, so that the readers can choose the most suitable one for their particular applications.

The remainder of this paper is organized as follows. Section II categorizes various hybrid storage architectures. Section III analyzes different kinds of algorithms used in the hybrid storage system. Section IV compares the performance of different types of hybrid storage systems. Finally, Section V draws the conclusion and provides some future research directions.

II. HYBRID STORAGE ARCHITECTURE

Besides the basic tiering method and caching method mentioned previously, the hybrid storage architectures that consist of HDD and SSD devices, in the rest of this paper, the hybrid storage architecture refers to these storage systems, also have some special structures, such as the combination of tiering and caching method, and the HDD device acts as the write cache of the SSD device. In all, there are roughly 4 types of hybrid storage architectures, which are: 1) SSD caching method, 2) SSD tiering method, 3) SSDs as the combination of tier and cache, which is called hybrid method, and 4) HDDs with special purposes, e.g., HDDs are utilized as the cache or mirror of SSDs. Some of the research works about the hybrid storage architectures are classified and listed in Table 2. There are also some hybrid storage systems incorporating other types of NVMs into design consideration, which will be discussed at the end of this section.

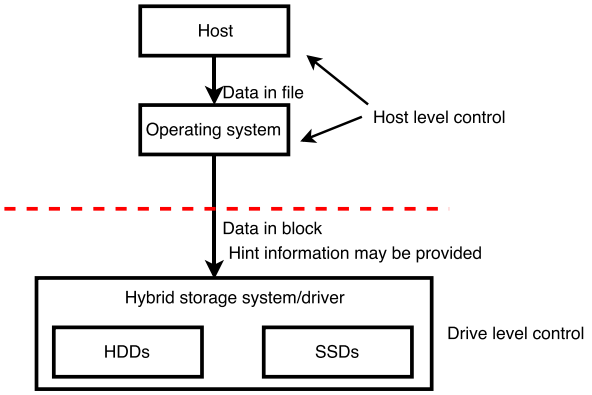


FIGURE 3. The difference between the host and device controlled policies.

A. SSD CACHING METHOD

To enhance the overall performance of a storage system, SSD is usually utilized as a cache between the DRAM and HDDs to keep the hot read data or the write data [26], [38], [43], [50]–[53]. Some of the storage architectures with SSD caching methods are shown in Table 3. The cache architectures are classified as host controlled and device controlled based on the location of the caching methods. In the host controlled caching architecture, the cache algorithm is normally implemented in the host side or operating system, and the data are usually in file format. In the device controlled cache architecture, the cache policy is usually utilized in the device itself and implemented in the firmware, and the data are usually in block. The difference between the host controlled and device controlled policies are shown in Fig. 3.

From Table 3, we can find that the common data flows inside the systems are as shown in Fig. 4. 1) If a read request arrives and its accessing data are located in SSD, the read

TABLE 3. Comparison of the hybrid storage systems with SSD caching method.

Literature	Cache policy	Data to SSD	Host or drive control	RAID or Deduplication	Background data movement	Simulation or protocol
Hot Random offloading [26]	Write back	random, hot	Host	No	Promote, demote	Protocol
Duplication aware SSD cache [27]	Write back	Hot	Drive	No	Promote	Simulation
SSD array as cache [28]	Write back	Hot Read, Write	Host	Yes	Flush, promote	Protocol
HFA-hint frequency based [29]	Write back	Hot Read, Write	Drive, hint	No	Demote, promote	Simulation
MOLAR [30]	Read only	Hot	Drive	No	Promote	Simulation
SLA hybrid store [31]	Read only	SLA, Hot	Host	No	Promote	Simulation
Lazy adaptive replacement [32]	Write back	Hot	Drive	No	Flush, promote	Simulation
FlashCache [33]	Write back	Hot Read, write	Host	No	Flush, promote	Simulation
NAND based disk cache [34]	Write back	Wear leveling aware	Drive	No	Flush, promote	Simulation
Caching less for better performance [35]	Write back	Hot	Drive	No	Flush, promote	Simulation
HEC [36]	Read only	Hot, Random	Drive	No	Pre-fetch	Simulation
EDM [37]	Write back	Hot	Host	No	Promote, demote	Protocol
Cache everywhere [38]	Write back	Cost-aware	Drive	No	Promote, demote	Simulation
AMC [39]	Write back	Data value	Drive	No	Promote, demote	Simulation
WEC [40]	Write back	Hot	Drive	No	Promote	Protocol
ETD-Cache [41]	Read only	Hot	Drive	No	Demote	Simulation
Regional popularity aware [42]	Write back	Hot Read, write	Drive	No	Promote, Flush	Protocol
SAR [100]	Write back	Hot	Host	No	Promote, demote	Protocol

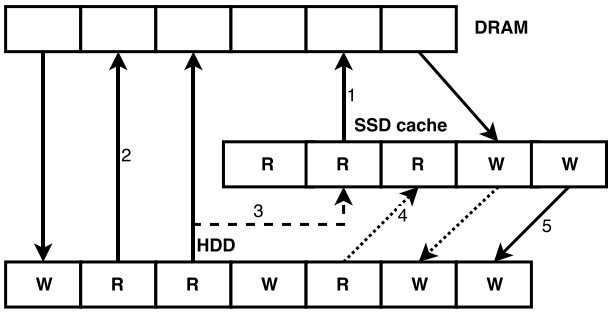


FIGURE 4. The data flows in the system with SSD as a cache to HDD.

request can directly access the data from SSD and complete the process. 2) If the accessed data are located in HDD, the data need to be read from HDD first, and then cached inside the DRAM. If the data are not hot enough, the process is completed. 3) If the data accessed are identified as hot by the hot data identification algorithm, the data migration algorithm moves the data to the SSD device. 4) Background data migration can occur for the data identified as hot. 5) The write data can be flushed to the HDD from SSD cache in background. From Table 3, we can find that the common data flows inside the systems are as shown in Fig. 4. 1) If a read request is arrived and its accessing data are located in SSD, the read request can directly access the data from SSD and complete the process. 2) If the accessed data are located in HDD, the data need to be read from HDD first, and then cached inside the DRAM. If the data are not hot enough, the process is completed. 3) If the data accessed are identified as hot by the hot data identification algorithm, the data migration algorithm moves the data to the SSD device. 4) Background data migration can be occurred for the data identified as hot. 5) The write data can be flushed to the HDD from SSD cache in background.

Based on the methods to handle the incoming write requests, the storage architectures with SSD caching methods can be categorized into two types: 1) SSD is utilized as the cache for read request only, which is the read-only policy. 2) SSD can be utilized as the cache for both read and write requests, which is the write-back policy.

1) SSD AS READ-ONLY CACHE

In this type of storage architectures [30], [31], [41], [54], when a new write request arrives and its accessing data are not located in SSD, the data need to be recorded to HDD and the request is completed only when the recording is successful. If the access data are available in SSD, the data in HDD need to be updated, and the data in SSD may be discarded, or updated as well. Only when both of these updating or discarding processes are successful, the write request is considered as completed. As the write requests are directly passed to HDD in this architecture, the number of write operations to SSD is reduced, and the SSD lifespan is prolonged. Meanwhile, the cache space utilized by the write data are saved and the saved cache space can be allocated for the read data, which might improve the read performance. Besides that, the selection of read data to reside in SSD is also important to balance the number of SSD writes and overall system performance. If more data are selected to move into the SSD device, the SSD lifespan is reduced, and more garbage collection (GC) processes might be required when

the SSD capacity is nearly full. Although the wear-leveling techniques inside the SSD can help to reduce the number of GC processes, it is better to reduce the unnecessary write operations to SSD.

To reduce the unnecessary write operations, [30] proposes a method to check the data hotness based on the demotion counter and the proposed control metric, and migrate the hot data blocks to SSD. In [54], a heuristic file-placement algorithm is designed to improve the cache performance by considering the IO patterns of the incoming workload. Meanwhile, a distributed caching middleware is implemented at user level to detect and manipulate the frequently-accessed data. In [17], a hybrid structure that combines SSD and SMR devices is proposed, which utilizes SSD as the read-only cache to improve the random read performance. In [30], the number of demotions from DRAM to SSD is recorded and utilized to predict the future accesses to SSD, and decide whether the data need to be recorded in SSD or not.

2) SSD AS A WRITE-BACK CACHE

There are two types of write cache, which are write-through cache and write-back cache. In the system with write-through cache policy, the write data are buffered to the cache and also to the disk, and it is only considered as complete when the data are successfully written to the disk. This policy is mainly utilized in the DRAM cache to avoid the data loss during power off as the DRAM is volatile memory, and improve the performance of read on write data in cache. Meanwhile, the system with write-back policy can write the data to cache first, and flush the data to disk at a later time, which can improve the write performance significantly in some cases. In a hybrid storage system with SSD as a cache, the write-through policy is seldom applied as the SSD is non-volatile memory, which can keep the data safe even when the drive power is lost.

To enhance the write performance of the hybrid storage system, some research works [51], [55]–[58] utilize SSD as the cache for both read and write requests. The data operations in the storage architectures with write-back policy are shown in Fig. 4. In the architecture, if the write data are new, the data will be recorded to SSD. With the write-back policy, these write data might be flushed to HDD at a later time. However, because of the later flush, the write data in HDD can not be updated simultaneously when it is written to SSD, which may cause data synchronization problem. Period write data flushing might be applied to flush the write data to HDD to improve the data synchronization.

Normally, utilizing SSD as a write-back cache can improve the performance of a hybrid storage system. However, when the SSD is nearly full, GC process is required to clean the invalid data before that the new write requests comes in, which lowers the performance for write requests, and the overall hybrid system performance might degrade. If the workload is write intensive with high IOPS, the number of requests to HDD might be large if the re-access ratio is small, the disk may have limited number of idle periods, and the durations of idle periods will be short. The write data might not be able to be flushed completely to HDD during these short idle periods. Then when the space of SSD is nearly full, the flushing of write data to HDD may further downgrade the system performance. Meanwhile, due to the non-volatility of SSD device, data can be kept in SSD for long time without demoting before the space is full, the system performance can be improved by applying the write-back cache policy. However, if the SSD fails, the write data inside might be lost. To overcome the SSD failure problem, the write data inside SSD are flushed to HDD as soon as possible at the cost of reducing the system performance. Furthermore, the SSD lifespan is another problem as the number of write operations to SSD is much larger compared with the storage architecture that uses SSD as read-only cache.

There are some research works [14]–[16], which integrate SSD and the SMR drive as a hybrid storage system to further increase the storage capacity whereas maintaining the accessing speed. The SSD is utilized as the write cache to keep the random write data, and leaves the sequential write data to be handled by the SMR drive. At a later time, the random write requests are combined and written to the SMR drive in sequence. With this, the number of random write requests to the SMR drive is reduced, as well as the number of read-modify-write operations. The overall performance can be significantly improved.

Industry players apply RAID techniques to SSD cache to improve the reliability of written data. For example, RAID-1 is employed in EnhanceIO [59], RAID-4/5 is employed in LSI ElasticCache [60] and HP SmartCache [61]. In research area, RAID techniques are also applied to SSDs to improve the data reliability and save the total cost. In [28], a new hybrid storage architecture is designed that employs low-end SSDs to construct a RAID architecture, which is utilized as the cache to improve the data reliability. The new architecture combines the advantages of data reliability of RAID system and performance of log-structured approach to achieve better performance per dollar and lifetime per dollar compared with high-end SSDs.

Some researchers [34], [35] divide the SSD space into read and write regions to improve the system performance with near balanced read and write requests. The read region is only used to store the incoming read data, meanwhile, the write region is only used to record the incoming write data. The data flows for these kinds of storage architectures are shown in Fig. 5. As shown in the figure, there are mainly 6 special operations in this architecture: 1) The write data are kept in the write cache region. 2) When the write request needs to access the data located in the read region, there are two options to handle this situation: If the cache is DRAM, the write request will directly update the data located in the read region, and then moves the data from read region to write region. If the cache is SSD cache, the write request will keep its new data to the write region, and mark the original data in SSD as invalid. The invalid data can be cleaned at a later time. 3) If the data in write region are flushed to the disk, and the

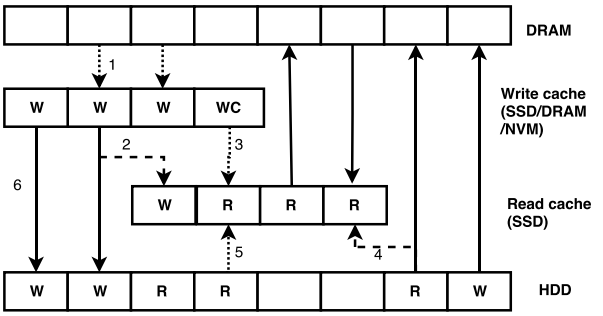


FIGURE 5. The data flows in the system with SSD as a read and write split cache to HDD.

data are frequently accessed by read requests, then the clean data can be moved to the read region. 4) The data accessed by read requests can be allocated to the read region. 5) The read data located in the HDD can be moved to the read region if they are identified as hot. 6) The data in the write region can be flushed to disk through background process. With this kind of cache architecture, the cache replacement algorithm can be improved, and the GC process can be more efficient. Thus the overall system performance is improved. However, if the incoming workload is read-intensive or write-intensive, the corresponding write cache region and read cache region will be wasted, and the overall system performance might degrade.

Write updating to SSD influences the data in the original block and makes the them become invalid. GC process is needed to clean these invalid data blocks, which requires extra SSD space. In [35], a new over-provisioned region is split out and used for the GC process. The split ratios of the read, write and over-provisioned region are analyzed through modeling to provide optimal system performance.

The system performance, especially the write performance, can be enhanced by utilizing SSD as the write-back cache. However, because the huge number of write requests to SSD, the SSD lifespan is reduced significantly. Meanwhile, the number of GC processes required in SSD may also increase. Keeping in mind the end goal to utilize the advantage of SSD write-back cache without reducing the SSD lifespan and increasing the number of GC processes excessively, numerous algorithms [27], [32], [41], [62] and architectures [36] are proposed to reduce the number of write requests to SSD. In [27], a duplication-aware write handling strategy is used to filter out the duplicated write requests, which saves the number of write operations to SSD device. In [32], the data replacement is done as late as possible, which saves some of the write operations caused by the data promotion and demotion. Despite the fact that it might miss some new generated hot data, the SSD lifespan is saved and the hot data with longer re-access interval may be kept.

To make the data allocation and migration more precise, the host side information might be utilized to check the hotness of the new coming data, as it might contain more valuable information. For instance, in [63], a multilevel cache hierarchy is proposed and the cache allocation depends on the host side hint information. In the system, the hint information, including access frequency and pattern to different range of storage space by each client, is provided to improve the cache performance. Cache partition for different range of storage space is allocated, and the hotter partition is allocated to the higher speed cache to improve the system performance. Meanwhile, the historical hint information can likewise be utilized to enhance the system performance, e.g., [29] uses historical hint information to enhance the storage system performance. The proposed system consolidates existing works, e.g. Hint-K [101], Demote [102], and Promote [103], together to enhance the performance of the multi-level cache.

B. SSD TIERING METHOD

Tiering techniques divide the data into hot and cold categories, and store them in SSD and HDD respectively. The data stored in SSD tier is permanent instead of temporary as in SSD caching, and there is no duplicate copy in HDD. A portion of the research works about SSD tiering strategy are listed in Table 4. From the table, we can observe that the tiered storage architectures can be isolated into two classifications, which are host controlled tiered structure [67], [69], [70], [73], [76], [81], [82] and device controlled tiered structure [66], [71], [72], [74], [75], [77]. In the host controlled tiered structure, the data movements are controlled by the system host by using the host side information, whereas the device controlled tiered structure controls the data movements by analyzing the properties of the current incoming data and the history data.

One of the tiered structures containing SSD tier and HDD tier is shown in Fig. 6. The data movements inside the tiered structure include: The data allocation, which is actively performed to specifically designate the data to suitable tier, which is shown as operations 1 and 2; The data migration, which is performed in background to relocate the data between the two tiers through steps 3 and 4 in Fig. 6. The data allocation and migration are typically controlled by the host in host controlled tiered structure or the device itself in device controlled tiered structure.

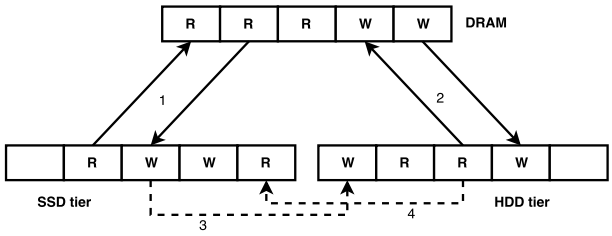


FIGURE 6. The data flows in the system with SSD as a tier.

In the host controlled tiered structure, the choice of which tier to store the approaching data is made by the host through analyzing the host side information. A linearly weighted formulation [70] is applied at the operating system level to characterize the workload access pattern to enhance the overall performance. GreenDM [67] combines the SSD

TABLE 4. Comparison of the hybrid storage systems with SSD tiering method.

Literature	Host or drive control	Hot or cold	Read or write	Random or Sequential	SLA and QoS	Data migration	Data allocation	Simulation or protocol
Hystor [66]	Drive	Yes	Yes	No	No	Yes	Yes	Protocol
GreenDM [67]	Host	Yes	No	No	No	Yes	Yes	Protocol
Hybrid store [68]	Drive	Yes	No	Yes	No	Yes	Yes	Protocol
Tiera [69]	Host	Yes	No	No	No	Yes	Yes	Protocol
Workload characterization [70]	Host	Yes	Yes	No	No	Yes	Yes	Protocol
E-hash [71]	Drive	Yes	Yes	No	No	Yes	Yes	Simulation
Tier warming [72]	Drive	No	No	No	No	Yes	Yes	Both
Extent based dynamic tiering [73]	Host	No	No	Yes	No	Yes	Yes	Protocol
Adaptive data migration [74]	Drive	Yes	No	No	No	Yes	Yes	Protocol
Efficient QoS [75]	Drive	No	Yes	Yes	Yes	No	Yes	Both
Balancing fairness and efficiency [76]	Host	No	No	No	Yes	No	Yes	Both
Automated storage tiering [77]	Drive	Yes	Yes	No	Yes	Yes	Yes	Model
FDTM [78]	Drive	Yes	Yes	Yes	No	Yes	Yes	Simulation
Optimal disk allocation [79]	Host	Yes	No	Yes	No	Yes	Yes	Simulation

devices and HDD devices in a storage system, and a block level interface is provided to allow data migration between SSD and HDD. GreenDM is adaptable to different kinds of incoming workloads by providing some valuable tunable parameters.

In the device controlled tiered structure, the decision of data movements is made by the device itself, which is the tier module implemented in the firmware, through analyzing the historical access information and the properties of the incoming workload. Hystor [66] analyzes the data access pattern, especially the access locality to find the critical access data. It saves the critical data to the SSD tier to enhance the system performance. Meanwhile, write data are also allocated to the SSD tier to improve the write performance. Although the write performance is improved, the SSD lifespan is reduced and the number of GC processes may increase. Hybrid-store [68] utilizes the regional access frequency information through analyzing the historical access information to distinguish the hot data and move them to SSD tier to improve the overall performance.

Data allocation and migration in a tiered storage system are usually determined by the data hotness. Notwithstanding, the device status and the data properties are not considered in these sorts of systems. In [83], a measurement-driven mechanism is applied for the data relocation between different tiers. In the system, the cold data are kept in slower tier. The hot data might be stored to different tiers in view of the data properties and the status of the devices in various tiers. For instance, the hot random data might be kept in SSD as SSD has faster random accessing speed. The hot sequential accessing data are kept in HDD to save SSD bandwidth, as HDD has similar sequential performance compared with SSD. In order to save the SSD lifespan and reduce the number of GC processes in SSD, the hot data identification and data migration algorithms need to be carefully designed. The trade-off between the system performance and the size of written data to SSD needs to be carefully considered.

To save the power consumption of hybrid storage system, [71] develops a novel architecture that consists of one SSD and multiple HDDs, which is called E-HASH. The architecture fully utilizes the advantages of random access of SSD and sequential access of HDDs to improve the system performance, whereas minimizing the overall cost. Meanwhile, the lifespan of SSD is also prolonged.

Tier warm-up is also deployed to improve the tiered storage system performance. In [72], the user access patterns over long time-scales are automatically learned and the intensity of the future user works can be predicted. Based on the knowledge of the tier capacity and the performance differences, a schedule can be made to decide when to start and stop the background system works. The data required by the background system works or the user workloads can be pre-filled to the faster tier to improve the overall system performance during the transition period.

C. SSD HYBRID METHOD

Both the SSD caching method and SSD tiering method have their own focal points and inconveniences. In tiering method, the frequency of data location optimization is extremely low, usually less than 1 time per hour, or even per day. At the point when the hot data region changes, it needs a considerable measure of time for the system to move the hot data to SSD to enhance the future system performance. In caching method, the data location is optimized in short time if the hotness of the data are changed and the system performance in subsequent accesses can be improved. However, the frequent data movements between HDDs and SSDs increase the load of devices, and might influence the overall system performance. To eliminate the restrictions of the caching method and tiering method, the storage architectures in [89] and [92]–[94]

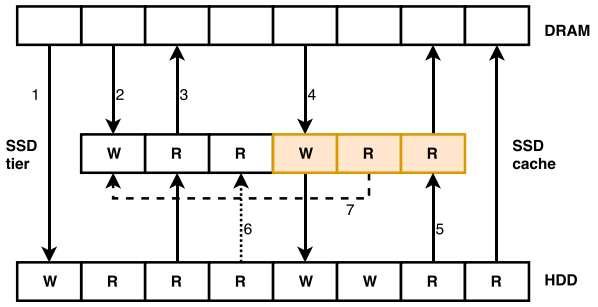


FIGURE 7. The data flows in the system with SSD tiering and caching methods.

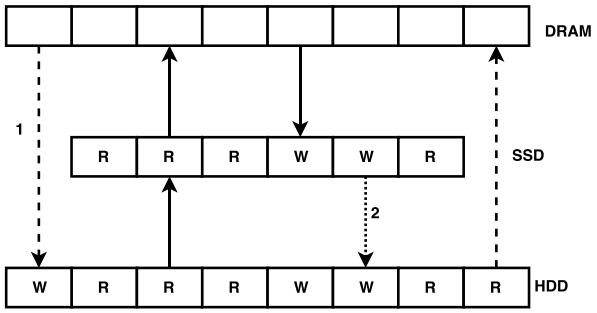


FIGURE 8. The data flows in the system with HDD as the write cache of SSD.

combine the tiering method and caching method together, which is the SSD hybrid method to utilize the SSD as both tier and cache to improve the overall system performance. A simplified data work flow for these kinds of storage architectures are shown in Fig. 7: 1) The write data are directly recorded to the HDD if the data are not inside SSD. 2) The write request updates the write data in SSD tier region if the data are hot and located in SSD tier. 3) The read request reads data from SSD tier region if the data are located inside SSD. 4) The write request buffers the data to SSD cache region if write-back policy is applied. 5) The read request reads data from HDD and buffers the data to SSD cache. 6) The hot read data are migrated to the SSD tier region in background. 7) The data movements between the cache and tier region are an optional operations.

Reference [89] consolidates the tiering and caching methods together in a hybrid storage array containing SSDs and HDDs. An allocation proportion is defined to adjust the number of SSDs in the tiering and the caching structure. The system performance, especially for the write intensive workloads is enhanced compared with the one without combination. Reference [90] utilizes the data hotness and the load status of the SSD and HDD devices to perform the data allocation and migration. The parallelism execution of all the devices are fully utilized to improve the overall performance. In [91], a hybrid storage aware flash translation layer (FDTL) is introduced to allocate the data to different levels of devices. The data allocation is based on the polices of utility maximization and energy consumption minimization. It combines the advantages of the sequential access speed of HDD and random access speed of SSD, which improves the IO performance and reduces the energy consumption.

D. OTHER STRUCTURES WITH SPECIAL PURPOSES

Besides storage architectures with SSD as tier and/or cache, there are some other types of storage architectures. For instance, some of the hybrid storage architectures employ HDD as the write cache for SSD [95], [96]. The operations in this kind of storage architecture are shown in Fig. 8. Compared with other types of storage architectures, there are 3 special operations: 1) The original data, which are the first copy of data is located to the SSD, whereas the update data will be located to HDD. 2) The write data might need to be flushed to HDD in background. With this type of architecture, the SSD lifespan is prolonged and the number of GC processes required is significantly reduced as the write update is handled mainly by HDD devices. The only drawback is that some of read requests might need to access both the HDD and SSD to get the complete required data.

I-CASH [95] is a new type of storage architecture that intelligently couples array of SSDs and HDDs. In the architecture, the SSD keeps the write data, called reference blocks, and HDD records the difference between the new coming data and the reference blocks, which is called delta. Read request is handled through the integration of the reference blocks and the deltas, and write request only needs to update the log of deltas, and thus the write update to SSD is minimized. With this architecture, the write performance is improved, and the lifespan of SSD is saved. However, the data might be lost if the delta is not saved to HDD properly. Thus the delta data in the DRAM should be flushed to disk frequently, and the flush interval is tuned based on the number of write updates that happened in the history.

A hybrid storage design called Griffin [96] uses HDD to cache write data to SSD. The write requests to SSD are logged into the HDD in sequence, and migrated back to SSD when the system is not busy. With such kind of design, the number of writes to SSD is reduced and thus the SSD lifespan is prolonged without significantly reducing the read performance.

In [97], the SSD space is split into data zone and delta zone, the data zone is used to store the new coming data, and the delta zone is used to store the difference between the history data and the new updated data. With the proposed mechanism, the small write penalty is reduced and the SSD lifespan is prolonged significantly.

To improve the data reliability and synchronization, one storage architecture [98] builds a SSD RAID0 storage system with multiple SSD devices, and deploys an HDD with identical space as the mirror to the SSD RAID. The proposed storage system can improve the data synchronization by lowering the IO performance a bit. Besides that, a multi-tier file system (TierFS) [99] is proposed to protect the stored data continuously, which keeps an extra copy of the updates

demoting to the lower tier until the update data backup is successful.

E. HYBRID STRUCTURES USING VARIOUS NVMs

As mentioned in Section I, besides NAND Flash, there are other types of NVMs. Some of them have better performance, longer endurance, and lower power consumption compared with NAND. However, their price per GB is much higher than NAND Flash. These kinds of NVMs are often utilized as a storage layer or a cache layer in modern hybrid storage systems to improve the performance cost ratio. In early stage, NVM is deployed as a cache to the HDD devices [27], [104], [105]. In [105] and [106], the NVM and NAND flash memory are both employed as the cache for HDD to improve the energy efficiency and overall system performance. In [27], NVM is added to improve the power management of the disk through four aspects: 1) extending the idle periods, 2) storing the read-miss content in the cache, 3) spinning up the media only when NVM cannot serve the requests, and 4) write throttling. In [107], the potential of PCM in storage hierarchy is evaluated. The tiering method and caching method are explored by modeling a hypothetical storage system consisting of PCM, HDD, and Flash to offer the best performance within cost constraints by identifying the combinations of different types of devices.

The higher performance NVM is also implemented as a cache for NAND flash [10]–[13]. In [10], the data generated by flash translation layer are stored in NVM, which prolongs the Flash memory lifespan. In [12], the combination of DRAM and PCM is deployed as a cache to SSD. A data migration policy is proposed to save the power consumption and enhance the SSD endurance. Meanwhile, the log information of NAND is stored in the PCM [13] to support for in-place updating and minimize the number of write updates to SSD, which reduces the Flash erase operation and prolongs the SSD lifespan.

Tiered memory system that utilizes NVM and DRAM is another type of tiered storage architectures. In [8] and [9], storage class memory (SCM) is deployed together with DRAM to form a tiered memory system, which leverages the advantage of capacity and data loss avoiding when power failure for SCM, and the advantage of access speed of DRAM. In [108], a new hybrid hardware/software solution that integrates PRAM and DRAM in the storage system is proposed. The write data reliability for PRAM is ensured by applying wear leveling uniformly through all the PRAM pages by considering the information of write frequencies.

III. ALGORITHMS UTILIZED IN HYBRID STORAGE SYSTEM

The algorithms and policies applied in the hybrid storage systems actually determine the performance of the overall storage system. In this section, the most important algorithms, such as data allocation, hot data identification, data migration, and scheduling algorithm are surveyed. For easy of representation, we list most of the factors considered in these algorithms in Table 5, where access frequency and interval are the most important two factors in hot data identification and data migration algorithms. Meanwhile, some of the algorithms also take the factors in Table 6 into consideration, which will not be discussed in detail.

A. DATA ALLOCATION

Hybrid storage system needs to decide which storage devices to handle the new coming data. Especially for the write data, the system needs to allocate the data to proper storage devices to fully utilize the storage resources. We call this problem as resource (data) allocation problem [76], [79], [82], [84], [105], [113]. In [79], the data allocation mechanism is formulated as a multiple choice knapsack problem. The optimal allocation solution can be found through multiple-stage dynamic programming. To maximize the system utilization, bottleneck aware allocation [76] is designed and developed to automatically choose the allocation ratio between clients with different bottleneck. To exploit parallelism between the HDD and SSD devices and provide lower response time, a space allocation policy [84] is designed to find the Wardrop equilibrium by utilizing initial block allocation and migration. Based on the allocation policy, the system can achieve load balancing adaptively across different levels of devices. A smart controller for disk cache [105] is employed to distribute the IO requests between NAND and PCM, where fine-granularity accesses are directed to the PCM, and coarse-granularity accesses are directed to flash memory to fully utilize their advantages.

B. ADDRESS MAPPING ALGORITHM

The address mapping between the SSD cache/tier and the HDD is necessary for the data access in hybrid storage system. Unified address mapping between the logical and physical block groups [91] can be applied to the physical/logical address translation. In the system, the address translation map is important and can not be lost, so the translation map is stored into the HDD.

In [114], the addresses of physical data blocks in the secondary storage and addresses of logical blocks corresponding to the physical blocks are stored in the mapping table. Each translation entry contains a value called current level, which is a function of the access frequency and the logarithmic system time. A top-K tracking table with B-tree structure is maintained to keep the first K elements with highest value of current level. When a new mapping entry is created, whose current level will be compared with the lowest one in the top-K tracking table, if it is higher, the lowest one will be replaced by the new coming entry. The value of K is selected based on the memory size allocated to the mapping table.

In [115], the performance difference is considered for different kinds of mapping tables under different run-time or load conditions. An adaptive address mapping with dynamic run-time memory mapping selection method is applied to optimize the mapping performance. A memory controller is deployed to monitor the online performance of

TABLE 5. Two aspects need to be considered during designing of hot data identification algorithm.

Items	Description	Typical algorithms
Access frequency	The number of times a data block is referenced	LFU (least frequently used) [109], GDSF (Greedy-Dual Size Frequency) [110]
Access interval	The time duration that a desired data item being accessed and re-accessed	LRU (least recently used) , MRU (most recently used), LFUDA (LFU with dynamic aging) [109], LRU-K(least recently used k) [111], GDS [112]

TABLE 6. The other workload properties need to be considered during storage system design.

Items	Description
Data size	Generally, only data with small size is required to move to higher tier. The small degree depends on the read/write speed of SSD and HDD, and the migration speed between them, etc. For example, large-size sequential data is usually unnecessary to migrate.
Cache total vs remaining size	The cache size decides how many hot data can be stored in cache. Hence it decides the threshold of hot degree
Device total vs remaining bandwidth	The bandwidth decides if the migration is proper at current time. An approximated function may be built to predict the remaining bandwidth with respect to R/W ratio, IO intensity, etc.
R/W ratio	As the R/W access time and pattern are different, this ratio may give different performance in some scenarios, e.g., the write amplification and GC in SSD and SMR.
R/W granularity and IO intensity	The value represents the ratio between an IO (both R/W) data amount and the fixed data block size. The average R/W granularity is only considered during certain accessing period. Usually, the data are more important to the user when the value of the data is larger.
Data correlation	One data may be related to another, i.e., Data access to certain data blocks within certain time period may have some access patterns, and other blocks / time periods may have similar pattern, and hence, they are associated. This value can be used for predication
IO range vs amount vs distribution	IO distribution represents the statistical accessing information, including the accessing address range and the accessing frequency in a given accessing period
Grain size	The minimum size for each page/block to be replaced/migrated
Tier contrast vs compensation (Device value)	It values the difference between two different storage tiers/caches for direct data migration, including device status, accessing speed, etc.
Others	Data loss/error, etc

current memory mapping and dynamically adjust the memory mappings at run-time based on the observed performance.

C. HOT DATA IDENTIFICATION AND DATA MIGRATION PROCESS

The hybrid system performance is heavily influenced by the selected caching policies, especially the data identification and migration algorithms. Numerous research works have focused on improving the hot data identification accuracy and migration efficiency, some of which are shown in Table 7. These hot data identification algorithms are mainly based on factors shown in Tables 5 and 6.

1) ACCESS FREQUENCY

Some of the hot data identification algorithms [87], [116] identify the frequently used data by counting the number of accesses. In [116], the hot data is identified through a probabilistic modeling of the incoming data, which can reduce the ratio of false detection of cold data as hot data.

2) ACCESS FREQUENCY AND RECENCY

Besides the access frequency, some algorithms [27], [74], [78], [85], [117] also utilize the access recency, which is last accessing time of the data to identify the hot data. Temperature aware caching (TAC) [50] checks the hotness of the data by analyzing the temperature statistics, which is the access frequency and the age of each region. A bloom filter [85] is applied to check the hotness of data through spatial locality and temporal locality. HotDataTrap [117] attemps statistical approach to avoid some cold data being inserted into the cache, which gives more chance for the hot data to be stored in the cache space. An improved adaptive cache replacement algorithm named D-ARC [27] is proposed to identify the hot data through both the recency and frequency of the incoming workload. D-ARC algorithm can improve the cache hit ratio, average I/O latency and the SSD lifespan significantly for certain types of workloads. Multiple independent hash functions [118] are applied to increase the correctness of hot data identification.

3) FREQUENCY/RECENCY WITH OTHER FACTORS

Some algorithms [26], [119] use other properties of the incoming workload to identify and migrate the hot data to SSD. Hot Random Off-loading method [26] checks the randomness and hotness of incoming data to determine the direction of the file migration. In [119], a block-level valuation model is applied with data value viewing from file-level to classify the data in the storage. In [120], a framework is proposed to quickly detect and predict the properties changing in the workload to make the migration process automatically. The detection and prediction are done through a clustering method that utilizes a Markov chain correlation algorithm.

To save the power consumption of tiered storage system, [121] proposes a power-aware multi-level cache policy,

which is used to identify and select data for migration. In a storage system, the service level agreement (SLA) is quite important for the QoS requirements by the user. How to efficiently explore the available storage resources to provide high QoS to customers is a quite challenging problem. In [122], a new approach is presented to meet different kinds of SLA requirements through data migration processes in varied environments. The data allocation and migration are decided by the SLA requirements and the temperature of the data, which is the access frequency of the data.

4) DATA MIGRATION

Together with the hot data, the data migration algorithms [39], [88], [121], [124], [125] also need to consider other factors, such as the incoming workload properties and the status of the devices. To improve the storage and retrieval performance, in [88], a metric is created to categorize all the levels of storage pools, which is applied to prioritize the data segments, and the data migration is performed based on the priority of the segments. In [124], the data migration policy is designed based on the data value computed through data intrinsic attributes and the prospective values. In [74], the migration deadline is also considered, and an adaptive data migration model is proposed to attemp to keep the hot data in SSD. The adaptive data migration is realized through IO profiling and look-ahead migration, where the IO profiling predicts and exploits the hot accessed area and the look-ahead migration moves the data, which will become hot in near future to SSD. In [126], offline algorithms are explored for the Flash cache by considering both the hit ratio and Flash lifespan. A multi-stage heuristic is designed to approximate the offline optimal algorithm, and provide the same optimal read hit ratio whereas reducing the number of Flash erasures significantly [125] by a bi-directional migration policy to balance the storage QoS and migration cost. The policy has two thresholds to determine the data promotion and demotion. In [78], a fully automatic migration process is proposed and a double thresholds method is introduced to only migrate the most suitable data blocks. The data migration can be done for both directions: low-end to high-end and high-end to low-end. An adaptive multi-level cache (AMC) replacement algorithm [39] is applied to adaptively adjust the cache blocks between different cache levels. When new request data are coming, AMC can dynamically determine the cache level to locate the incoming data by combining the selective promotion and demotion operations. With AMC, the cache resource management becomes more efficient, and multi-level cache exclusiveness is achieved.

D. CACHE SCHEDULING ALGORITHMS

1) CACHE SCHEDULING ALGORITHMS IN HDD

There are numerous types of scheduling algorithms utilized in HDD device to optimize the disk service time, including the seek time, rotational latency and data transfer time, such as Rotation Position Optimization (RPO) [127], Shortest Access Time First (SATF) [128], SCAN [129], etc. However, due to the fundamental difference between SSD and mechanical driven HDD, these conventional scheduling algorithms implemented in HDD are not suitable for SSD drive and hybrid systems.

2) CACHE SCHEDULING ALGORITHMS IN SSD

To fully utilize the SSD properties, an efficient scheduling scheme [130] for NCQ in SSDs is proposed to improve the system performance. In the scheme, the I/O service time for each command is estimated based on the status of the buffer as well as the physical characteristics of SSDs. The scheduler selects the next command to be executed based on the estimated service time. Some scheduling algorithms also consider the workload properties. In [131], a workload-aware budget compensation scheduling algorithm is utilized for the device-level request scheduling. The scheduler estimates the cost of GC contribution for each of the virtual machines in the SSD device, and compensates the budget for each virtual machine based on the estimated cost for performance isolation.

3) CACHE SCHEDULING ALGORITHMS IN HYBRID STORAGE SYSTEM

Based on the difference of the scheduling algorithms in HDD and SSD, new scheduling algorithms need to consider the properties of HDDs and SSDs, the combination of HDD scheduling and SSD scheduling algorithms can be one of the solutions for hybrid storage system. In [**?**], a hybrid disk-aware CFQ is proposed to optimize the I/O reordering in new hybrid storage systems. In [132], a Parallel Issue Queuing (PIQ) scheduling method is applied to increase the parallel execution of incoming requests. It attempts to combine the non-conflict requests into the same batch to avoid unnecessary access conflicts.

Scheduling algorithms may also consider the SLA from users. To guarantee the SLA, [133] proposes a scheduling algorithm to provide guaranteed service level objectives in a dynamic environment. The proposed algorithm can dynamically adjust the number of active hosts, which makes the system suitable for various kinds of workloads with different IOPS and throughput, whereas minimizing the overall system power consumption.

IV. PERFORMANCE COMPARISON

In this section, the performances of different kinds of hybrid storage architectures are compared and analyzed. A hybrid storage system simulator is designed and developed to conduct the quantitative comparison for different kinds of storage architectures, e.g. SSD as a read-only cache, SSD as a write back cache, and SSD as a high tier, etc. The comparisons are conducted based on the following perspectives: 1) The system performance, such as the overall response time. 2) The SSD lifespan, which is simulated as the number of data written and erased in the SSD device. 3) The energy consumption, which includes energy consumption caused by HDD and

TABLE 7. Comparison of hot data identification algorithms.

Literature	Region levels	Frequency	Recency	Hot levels	Host or drive control	Consider R/W
HFA [29]	One	Yes	No	Hot, cold	Drive	No
Hot random offloading [26]	One	Yes	No	Hot, cold	Host	No
Tiera [69]	One	Yes	Yes	Hot, cold	Host	No
Regional popularity aware cache [42]	Two	Yes	Yes	Hot, cold	Drive	No
Azor [65]	Two	Yes	No	Hot, cold	Drive	Yes
GreenDM [67]	One	Yes	Yes	Hot, cold	Host	No
Multiple bloom filters [85]	Multiple	Yes	Yes	Hot and cold	Drive	Yes
Hotness aware hit [86]	Multiple	Yes	Yes	Hot, warm and cold	Drive	Yes
Block level data valuation [119]	One	Yes	Yes	Specific value	Drive	Yes
Duplication aware cache [27]	One	Yes	Yes	Hot and cold	Drive	Yes
AMC [39]	One	Yes	Yes	Hot and cold	Drive	Yes
HotDataTrap [123]	One	Yes	Yes	Hot and cold	Drive	No
multi-hash-function framework [118]	One	Yes	Yes	Hot, cold	Drive	No

SSD. To make the comparison more meaningful and intuitive, we focus on the comparisons between the different SSD caching architectures and the SSD caching and SSD tiering polices in their basic forms. The comparisons are conducted under the same hardware settings and workload environment. Note that although our settings are simple, it can still reflect the insight of the complex systems with disk arrays, as the corresponding array can be virtualized as a storage pool.

This section is divided into three parts. Firstly, the simulation tool and the algorithms utilized are presented in detail. Then, the hardware specification and system settings are provided in details. Lastly, the performances, including the overall response time, SSD lifespan usage, and the energy consumption, are compared and discussed thoroughly for different caching policies and system architectures.

A. SIMULATION DESIGN AND SETUP

Conventionally, the hybrid storage system simulation is studied and analyzed by numerous research works. OGSSim [134] is one of the storage simulators to study the performance of different kinds of storage systems with generic devices and architecture layouts. Reference [135] compares the performance of different kinds of storage systems, such as tiered storage, distributed storage and normal HDD storage. StorageSim [136] is developed to simulate the performance of a multiple tiered storage system under different kinds of data placement policies. However, all these simulation tools only focus one or two aspects of the hybrid storage system, and lack of comparisons between the basic architectures, such as SSD caching method and SSD tiering method. To study the performance, such as response time, energy consumption and SSD lifespan, under different kinds of hybrid storage systems, a hybrid storage simulator is designed and developed.

Our hybrid storage simulator mainly contains three components, the HDD simulator, the SSD simulator and the hybrid controller, which is shown in Fig. 9. The hybrid controller is implemented differently under different storage architectures.

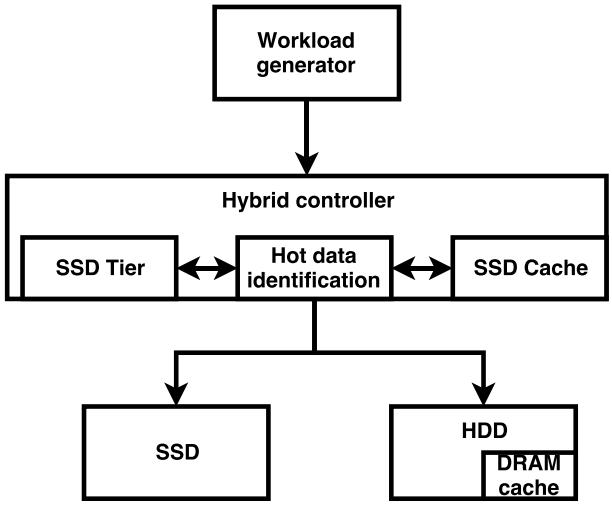


FIGURE 9. The overall structure of the hybrid storage system simulator.

The architecture with SSD tiering method contains a tier handler, which handles the data migration and request redirection, a hot data identification module, which detects the hot data. The architecture with SSD caching method consists of a cache handler, which has an LRU cache policy implemented. The architecture with SSD hybrid method consists of a tier handler, a cache handler and a hot data identification module. The data movements between the HDD and SSD inside the cache and tier components are processed by the tier and cache handler. In the cache handler, there are two types of cache polices implemented, i.e., read-only policy, and write-back policy.

B. SYSTEM SPECIFICATION

In the simulator, the specifications of HDD device are shown in Table 8, and the specifications of SSD device are shown in Table 9. The traces utilized in the simulation are listed in Table 10. There are 3 types of traces utilized in the simulation: 1) SPC1C trace [137] is an industry-standard

TABLE 8. The hardware specification of HDD in the system.

Property	Value
HDD capacity	600GB
HDD RPM	7200
HDD maximum seek time	11ms
HDD number of tracks	Around 300K
HDD sectors per track	2048 blocks average
HDD idle power	3.7W
HDD busy power	5.7W
HDD DRAM size	64MB
HDD DRAM cache line size	256 blocks
HDD cache policy	LRU

TABLE 9. The hardware specification of SSD in the system.

Property	Value
SSD read time	0.2ms
SSD write time	0.268ms
SSD erase time	1ms
SSD read power	2W
SSD write power	2W
SSD idle power	0.05W
SSD page size	256 blocks
SSD block size	256 pages
SSD cache policy	LRU

storage performance benchmark, and it represents the end user applications. 2) Financial trace [138] is the workload gathered through running the OLTP applications at two large financial institutions. 3) Web searching trace [138] is gathered from a popular search engine. The trace property re-access ratio is the number of requests that access the previous existing data over the total number of requests. The access frequency is the average number of accesses for each of the accessed range (the range size is set to be the same as the SSD page, which is 64K), which is an important factor to check the hotness of the data.

Other than that, the hot data identification threshold is set differently for different traces, e.g. 2 for SPC1C trace, 3 for web searching trace, and 5 for financial trace. All these parameters can be adjusted easily through the system settings.

C. PERFORMANCE COMPARISONS UNDER DIFFERENT CACHING POLICIES

In this section, the performances of different cache polices, including system response time, SSD lifespan usage and the energy consumption by HDD and SSD, are compared and analyzed under different kinds of workloads.

1) PERFORMANCE

When the free space of SSD is enough, the caching architecture with write-back policy has better performance than the read-only policy for the workloads SPC1C and OLTP, which contains certain amount of write requests. The performance comparisons for the SPC1C traces are shown in Fig. 10(a), and the performance comparisons for OLTP traces are shown in Table 11. The performance is similar for the read-intensive workload, such as web searching trace, which is shown in Table 12. The main reason for this is that the write requests are cached in the SSD device for the architecture with write-back policy, which lowers the response time of the write requests. Meanwhile the number of requests to the HDD device is reduced and the performance of read requests to HDD is improved as the load of HDD is reduced when the IOPS of the workload is large. As the SSD space is enough to cache the write data and hot read data, the flushing of write data to HDD only needs to be handled during the system idle time, the overhead of write data flushing is minimized.

As shown in Fig. 10(b), when the free space of SSD is not sufficient, the overall response time is increased for both the read-only policy and write-back policy for the workloads

TABLE 10. The different types of workloads and their properties.

Workload	Write ratio	Reaccess ratio	IOPS	Length	Access frequency
SPC1C trace	60%	15.71%	25-325	12K-155K	1.5733
Financial (OLTP) trace	78%	99.4%	336.6	4.413M	45.31
Web searching trace	0.1%	60.2%	334.78	1.055M	4.7031

TABLE 11. Performance comparison under different caching policies for financial trace.

Caching mode	Performance(s)		SSD lifespan		Energy(KWh)	
SSD free space	Limited	Enough	Limited	Enough	Limited	Enough
Read only	0.000300	0.000201	1294900	1496875	0.014494	0.014262
Write back	0.000687	0.000616	4157523	4113167	0.015396	0.014577

TABLE 12. Performance comparison under different caching policies for web searching trace.

Caching mode	Performance(s)		SSD lifespan		Energy(KWh)	
SSD free space	Limited	Enough	Limited	Enough	Limited	Enough
Read only	0.008755	0.000930	553215	77396	0.004682	0.003590
Write back	0.008755	0.000930	553439	77527	0.004683	0.003590

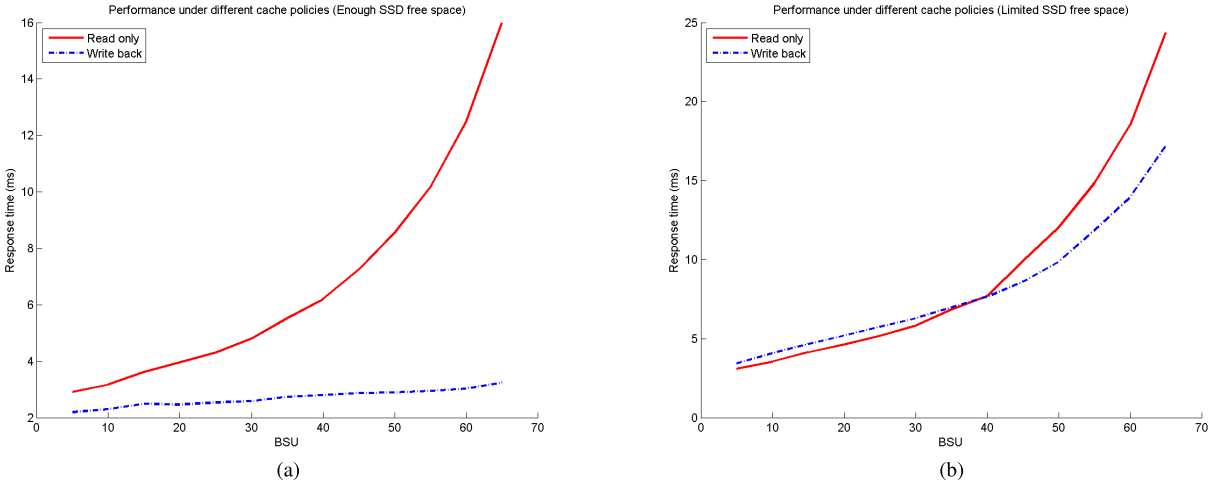


FIGURE 10. System performance under different caching policies for SPC1C traces. (a) Enough SSD free space. (b) Limited SSD free space.

with high IOPS. The increment for the write-back policy is significantly larger compared with read-only policy, as the write requests to SSD might cause write data flushing to HDD when the free space in SSD is not enough to store the new coming data, and the GC process inside the SSD device. The performance of the system with read-only policy also decreases, which is caused by the reduced number of hot data kept in the SSD cache, which might reduce the chance of read cache hit. Meanwhile, we also notice that there is a turn point around BSU 40, where 1 BSU equals to 5 IOPS. When BSU is small, the number of requests to HDD is small, there are enough idle periods for the cache to flush the write data to HDD, and the DRAM cache has clean space for storing the new coming write requests. Thus the read-only cache has better performance compared with write-back cache. However, when the BSU is large, the number of idle periods may not be enough to let the DRAM cache flush the write data, and the process of write data flush will increase the waiting time for the requests. In this circumstance, the write-back policy can keep the write data for a while, and have better performance compared with read-only cache policy.

In Table 11, the performances of systems with both caching policies for the financial trace are compared. We can notice that the performance with read-only policy is somewhat superior to the one with write-back approach, in light of the very high re-access ratio of the workload. Since majority of the requests can be dealt by the cache in SSD and DRAM, the disk will have enough idle periods to handle the flushed write data, which cleans the dirty data inside the DRAM cache. The cleaned space can be utilized to handle the future coming write requests. Thus the performance of hybrid storage system with SSD as the read-only cache is slightly better compared with write-back policy, on the grounds that the write access speed of DRAM is faster than SSD. In Table 12, the performances of the systems with both cache polices for web searching trace are compared, and we can see that the performance is almost same. This is on account of that the web searching trace is a read-intensive workload, which has very few write requests.

To reduce the overhead of data flushing and data copying between HDD and SSD, [26] identifies the hot data at the host side by analyzing the user level information, which provides more hints for the identification process. With the accurate hot data identification process, the system can minimize the cold data copying to SSD, and increase the chance of read cache hit. In [32], the lazy adaptive cache replacement algorithm can also reduce the chance of copying cold data to SSD, which also improves the read cache hit ratio and the overall system performance.

2) SSD LIFESPAN

In Fig. 11, the sizes of write data to the SSD, which affect the SSD lifespan most under different caching policies for the SPC1C traces are presented. We notice that the system with write-back cache policy writes more data to the SSD, this is the result of that the write requests are taken care of by the SSD devices. Meanwhile, the read data are likewise duplicated to the SSD cache from HDD for future read access. From the figure, we also notice that the size of write data to SSD is larger when the SSD free space is not enough to cover the incoming write data, this is caused by the GC process in SSD, which consists of the processes of reading data, erasing block, and writing data.

In Table 11, the number of SSD writes in the system with read-only policy is significantly less compared to the one with write-back policy. This is on the grounds that the write requests are mostly handled by the HDD device. The numbers of SSD writes for systems with read-only and write-back policies are similar for the web searching trace as shown in Table 12, which can be clarified by the high read ratio of the workload.

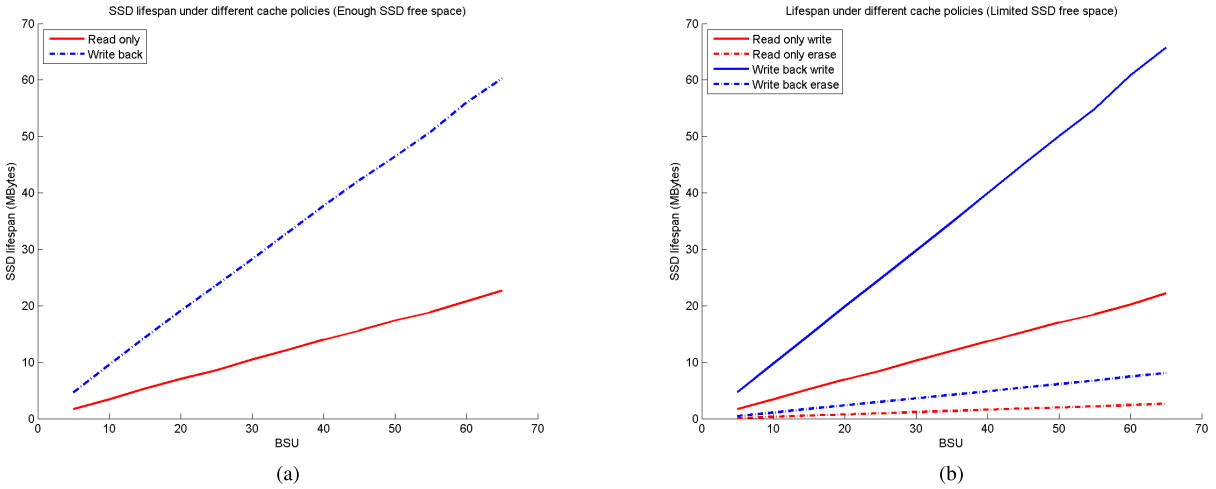


FIGURE 11. SSD lifespan under different caching policies for SPC1C traces. (a) Enough SSD free space. (b) Limited SSD free space.

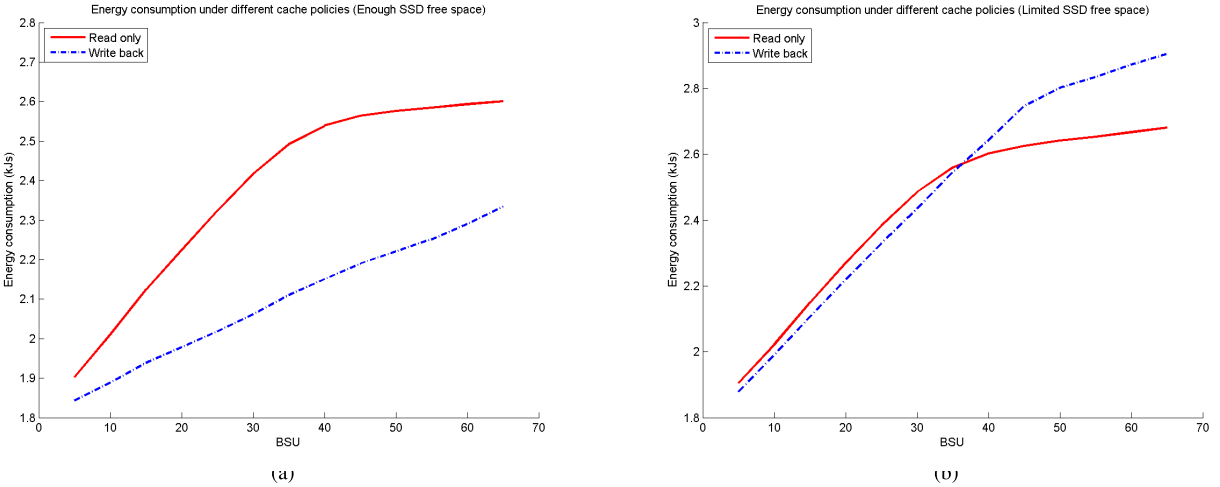


FIGURE 12. System energy consumption under different caching policies for SPC1C traces. (a) Enough SSD free space. (b) Limited SSD free space.

Minimizing the number of writes to SSD [36], [40] can prolong the SSD lifespan at the cost of decreasing the overall system performance. In [26], the SSD device only caches for the random write data, and the sequential write data are located to the HDD, which prolongs the SSD lifespan. In [27], before the write request coming to the storage system, deduplication process is conducted to remove the duplicated data, which can reduce the number of updates significantly when the workload contains lots of update requests. At that point, the SSD lifespan can be prolonged. In [32], due to the lazy adaptive cache replacement algorithm, the insertion and removing of cache entries from the SSD are more strict, which reduces the number of write operations to SSD and saves the lifespan of SSD.

3) ENERGY CONSUMPTION

The energy consumptions of the system with various caching policies for SPC1C traces are shown in Fig. 12. When SSD has enough free space, the system with read-only cache policy consumes more energy compared with the one applying write-back policy. This is due to that the write requests are mostly handled by HDD device for the system with read-only cache policy, which increases the HDD busy time and energy consumption. When SSD only has limited free space, the energy consumption is higher for the system with the read-only cache policy when the IOPS is low, but lower when the IOPS is high compared with the system with write-back policy. The principle explanation behind this is that when the free space in SSD is not enough, if the IOPS of the trace is higher, the busy period of the HDD device is similar for both read-only cache policy and write-back policy, as the write data need to be flushed to HDD under write-back policy. Meanwhile, the SSD device consumes more energy under write-back policy as it needs to handle more write requests. In Tables 11 and 12, we can observe that lower energy consumption of the system with read-only policy when the SSD free space is limited. This is on account of that the write requests need to be handled by SSD device first, and then

TABLE 13. Performance comparison under different system architectures for financial trace.

Caching mode	Performance(s)		SSD lifespan		Energy(KWh)	
SSD free space	Limited	Enough	Limited	Enough	Limited	Enough
SSD Cache	0.000300	0.000201	1294900	1496875	0.014494	0.014262
SSD Tier	0.000262	0.000198	1180351	1326725	0.014408	0.014245
SSD Hybrid	0.000219	0.000201	1282203	1496864	0.014358	0.014262

TABLE 14. Performance comparison under different system architectures for web searching trace.

Architecture	Performance(s)		SSD lifespan		Energy(KWh)	
SSD free space	Limited	Enough	Limited	Enough	Limited	Enough
SSD cache	0.008755	0.000930	553215	77396	0.004682	0.003590
SSD tier	0.020085	0.009704	78340	46041	0.004580	0.004137
SSD hybrid	0.011454	0.000930	559622	77396	0.004727	0.003590

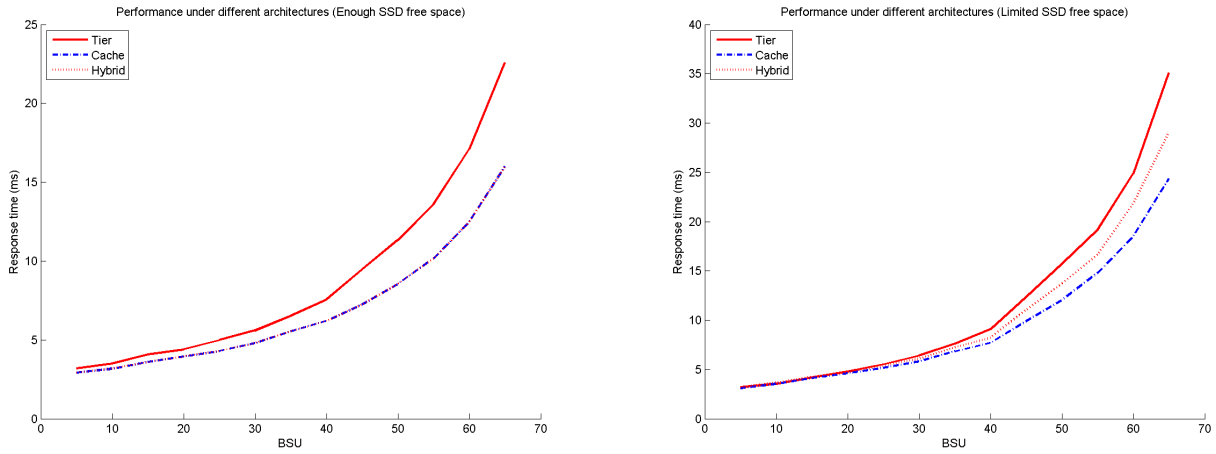


FIGURE 13. System performance under different storage architectures for SPC1C traces. (a) Enough SSD free space. (b) Limited SSD free space.

flushed to HDD in the system with write-back policy, which consumes more energy in SSD. When the free space of SSD is enough, the energy consumption is similar for both cache policies as the write data do not need to be flushed to HDD since the cache space is enough to cover the incoming write data.

D. PERFORMANCE COMPARISONS UNDER DIFFERENT SYSTEM ARCHITECTURES

In this section, we will discuss the performances of different storage architectures under different kinds of workloads. In order to make sure that the comparisons are fair, all the architectures apply read-only cache or tier policy in their system, where the SSD devices are only utilized to store the hot read data. The performance comparisons for the systems handling financial and web searching traces are listed in Tables 13 and 14, and the performances for SPC1C traces are shown in Fig. 13, 15 and 17.

1) PERFORMANCE

In Fig. 13, we can notice that the performance of the system with caching method is better than the one with tiering method and hybrid method, regardless of whether the SSD

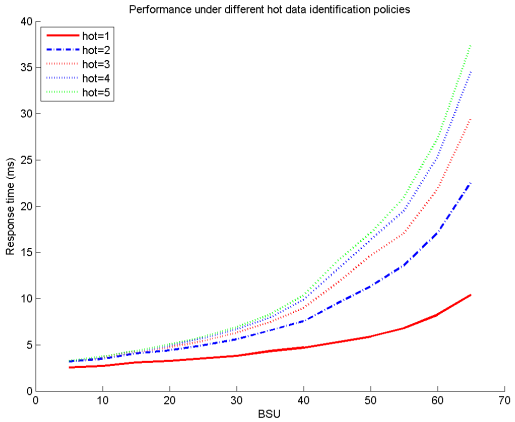


FIGURE 14. The system performance under different hot data identification policies for SPC1C traces.

free space is sufficient or not. This can be clarified by the properties of SPC1C traces. Due to the small re-access ratio of SPC1C traces, the number of hot data migrated to SSD is less compared with SSD caching method and the benefit of faster SSD read access can not be fully utilized with the tiering method as the read cache hit ratio is reduced. We additionally

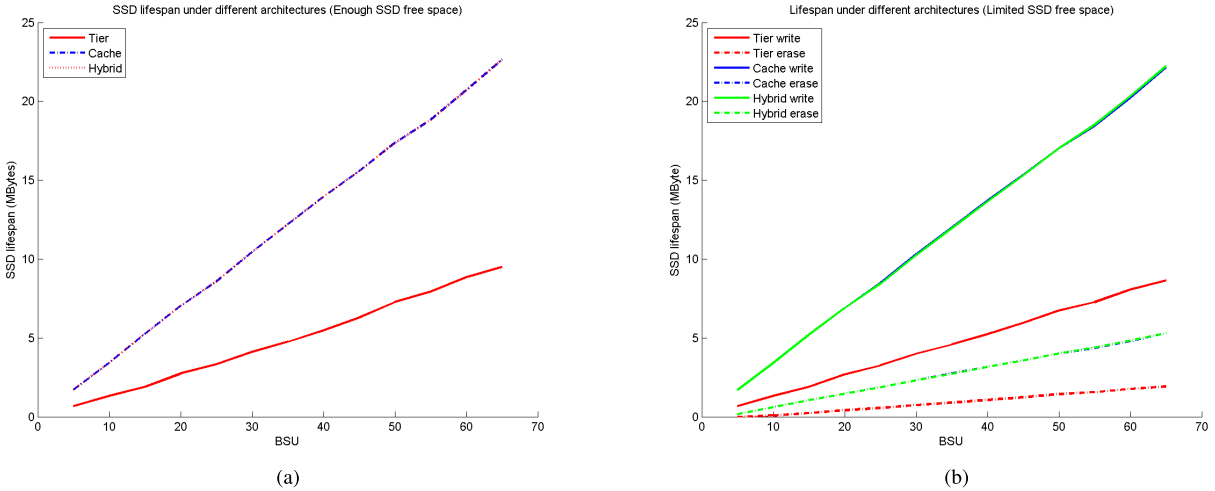


FIGURE 15. SSD lifespan under different storage architectures for SPC1C traces. (a) Enough SSD free spacey. (b) Limited SSD free space.

see that the system with the hybrid method has slightly better performance than the tiering method when the SSD free space is not enough. This is in light of the fact that the combination of tiering and caching method can fully utilize the benefit of caching method when the re-access ratio of the workload is smaller. Comparative things happened on the web searching trace, which also does not have high re-access ratio. However, when SSD free space is limited, if the re-access ratio of the workload is high, such as the financial trace, the performance of tiering method is better compared with caching method, which can be found in Table 13. This is on account of that the tiering method minimizes the data migration between the SSD and HDD and only the hot data are migrated to the SSD. To further improve the system performance, [90] considers the load status of the SSD and HDD devices during the data allocation and migration, which improves the system performance by parallel usage of the devices.

In a tiered storage system, the system performance is also highly affected by the hot data identification and data migration policies, which can be observed in Fig. 14. From the figure, it is easy to notice that the system performance decreases when the data hotness threshold increases. This can be explained by the fact that the number of data migrated to SSD is reduced when the hotness threshold is increased. The benefit of the faster accessing speed of SSD can not be fully utilized compared the one with lower hotness threshold. More read requests need to access data from HDD, which lower the overall system performance.

2) SSD LIFESPAN

The SSD lifespan usages under various storage architectures for SPC1C traces are shown in Fig. 15. From the figure, we notice that the SSD lifespan usage under the tiering method is vastly improved compared with the caching method and hybrid method. This is because the tiering method only moves the data to SSD when the data are identified as hot, whereas the caching method moves the data to SSD when they are read from HDD. In Tables 13 and 14, the SSD lifespan usage with financial trace and web searching trace are shown. We notice that the SSD lifespan usage under tiering method is always smaller compared with the caching method and the hybrid method for both of the traces no matter the SSD free space is enough or not. In order to improve the overall system performance, the load balance is also considered. In [90], the SSD lifespan is further utilized as the algorithm needs to maximize the parallelism of the devices, which requires more write operations to SSD when the load of HDD is very high.

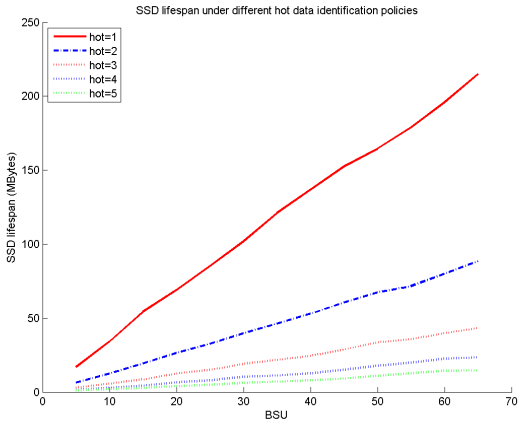


FIGURE 16. The SSD lifespan under different hot data identification policies for SPC1C traces.

In a tiered storage system, the SSD lifespan is also affected by the hot data identification policies, which can be shown in Fig. 16. From the figure, we notice that when the hotness threshold is large, the more SSD lifespan is saved. This can be explained by the fact that the size of hot data moved to SSD are reduced when the hotness threshold is large compared with the one having a smaller hotness threshold.

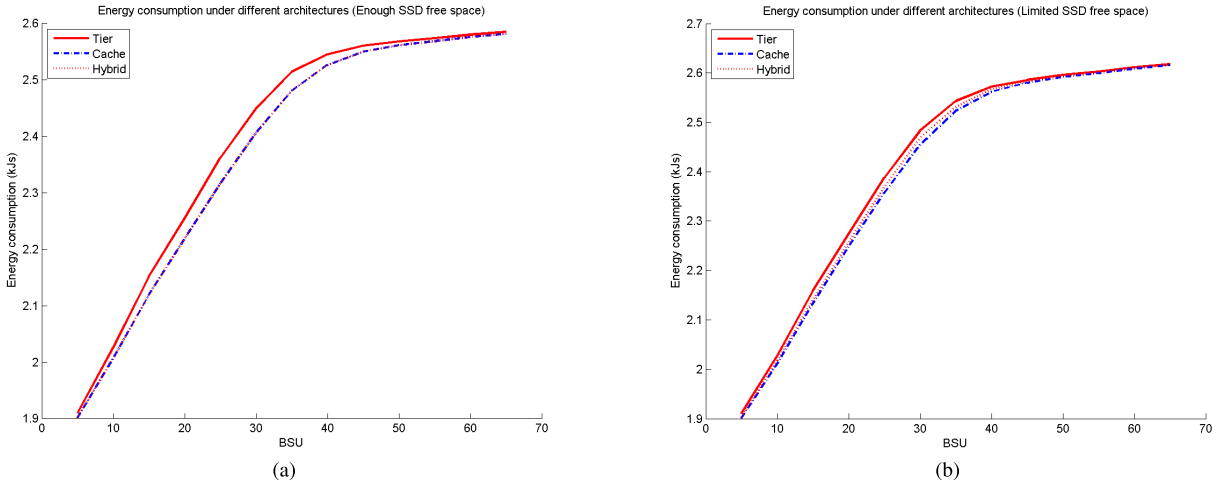


FIGURE 17. System energy consumption under different storage architectures for SPC1C traces. (a) Enough SSD free space. (b) Limited SSD free space.

3) ENERGY CONSUMPTION

In Fig. 17, the energy consumptions of the systems with various architectures are shown. From the figure, we can observe that the tiering method consumes more energy when the IOPS of the workload is smaller no matter the SSD free space is enough or not. This is on the ground of that the read request needs to access data from HDD several times before the data are migrated to SSD, whereas the data are directly copied to SSD for the first access in the system with caching method or hybrid method. When the IOPS of the workload is large, the load of the HDD is mostly full, and the difference between the energy consumptions of HDD for the tiering method and the other two architectures is smaller. Meanwhile, the tiering method has less SSD accesses and the energy consumption is less compared with the other two architectures. Along these lines, the overall energy consumptions of the three architectures are comparative when IOPS of the SPC1C trace is high. The energy consumptions of the different architectures for web search and financial traces are shown in Tables 13 and 14. From the tables, we can notice that the energy consumption of the system with tiering method is slightly smaller compared with caching and hybrid method when the SSD capacity is enough. This can be explained with the properties of these two workloads. The financial trace has very high re-access ratio, and the HDD is mostly in idle, the read access to HDD is mostly handled by the SSD cache or the DRAM cache in HDD, meanwhile, the higher number of data copying operations between HDD and SSD in the system with caching method may consume more energy. As the IOPS of web searching trace is high and re-access ratio is small, the HDD is busy for most of the time, thus the energy consumption of HDD is similar for all the three architectures. However due to the less data migrated to SSD in tiering method compared with the other two architectures, the energy consumption of SSD is much less. Therefore the overall energy consumption is less for tiering method.

V. CONCLUSION AND FUTURE DIRECTIONS

In this paper, we presented a comprehensive survey on the architectures and algorithms of hybrid storage systems. Right off the bat, we categorized and analyzed different hybrid storage architectures, including the tiering method, the caching method and the hybrid method. Furthermore, the algorithms and policies used in hybrid storage systems were studied, including data allocation, hot data identification, data migration, request scheduling and address mapping. At last, the advantages and disadvantages of some basic hybrid storage architectures were studied and compared quantitatively through simulation in several aspects, such as the system performance, SSD lifespan, and energy consumption, etc. Although it is generally difficult to provide quantitatively comparative results for all the systems in this survey due to the system complexity with large number of different algorithms in diversely hybrid structures, the simulated comparative results can still act as a basic benchmark to analyze the performance of different systems in the survey.

Research works on integration of the tiering method and caching method to improve the performance of the storage system are promising. However, the current integration techniques are only done at a basic combination level, and the impact of the integration is not analyzed in detail. At the same time, artificial intelligence and machine learning techniques have shown the capabilities in many industrial applications. One possible future direction is to investigate the integrated platform with those techniques for better adaptivity to dynamic workload and wider applications to various scenarios with tighter performance, reliability, cost, lifespan and energy requirements.

ACKNOWLEDGMENT

The authors would like to thank Dr. Jie Yu for his discussion on an earlier version, Robin O'Neill and David Chan for

their review, and Grant Mackey for his corrections on some grammar issues of this manuscripts.

REFERENCES

[1] V. Turner, J. F. Gantz, D. Reinsel, and S. Minton, "The digital universe of opportunities: Rich data and the increasing value of the Internet of Things," *IDC Anal. Future*, Apr. 2014.

[2] (2017). *Flash Memory—From Wikipedia, the Free Encyclopedia*. [Online]. Available: https://en.wikipedia.org/wiki/Flash_memory

[3] (2017). *Solid State Drive—From Wikipedia, the Free Encyclopedia*. [Online]. Available: https://en.wikipedia.org/wiki/Solid_state_drive

[4] N. Muppalaneni and K. Gopinath, "A multi-tier RAID storage system with RAID1 and RAID5," in *Proc. 14th Int. Conf. Parallel Distrib. Process. Symp. (IPDPS)*, 2000, pp. 663–671.

[5] H.-S. P. Wong *et al.*, "Phase change memory," *Proc. IEEE*, vol. 98, no. 12, pp. 2201–2227, Dec. 2010.

[6] Y. Huai, "Spin-transfer torque MRAM (STT-MRAM): Challenges and prospects," *AAPPS Bull.*, vol. 18, no. 6, pp. 33–40, 2008.

[7] H.-S. P. Wong *et al.*, "Metal–oxide RRAM," *Proc. IEEE*, vol. 100, no. 6, pp. 1951–1970, Jun. 2012.

[8] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," *ACM SIGARCH Comput. Archit. News*, vol. 37, no. 3, pp. 24–33, 2009.

[9] K. A. Bailey, P. Hornyack, L. Ceze, S. D. Gribble, and H. M. Levy, "Exploring storage class memory with key value stores," in *Proc. 1st Workshop Interact. NVM/FLASH Oper. Syst. Workloads (INFLOW)*, 2013, pp. 4:1–4:8.

[10] C. W. Smullen, J. Coffman, and S. Gurumurthi, "Accelerating enterprise solid-state disks with non-volatile merge caching," in *Proc. IEEE Int. Conf. Green Comput. (IGCC)*, Aug. 2010, pp. 203–214.

[11] N. Lu, I.-S. Choi, S.-H. Ko, and S.-D. Kim, "A PRAM based block updating management for hybrid solid state disk," *IEICE Electron. Exp.*, vol. 9, no. 4, pp. 320–325, 2012.

[12] M. Tarihi, H. Asadi, A. Haghdoost, M. Arjomand, and H. Sarbazi-Azad, "A hybrid non-volatile cache design for solid-state drives using comprehensive I/O characterization," *IEEE Trans. Comput.*, vol. 65, no. 6, pp. 1678–1691, Jun. 2016.

[13] G. Sun, Y. Joo, Y. Chen, Y. Chen, and Y. Xie, "A hybrid solid-state storage architecture for the performance, energy consumption, and lifetime improvement," in *Emerging Memory Technologies*. New York, NY, USA: Springer, 2014, pp. 51–77.

[14] W. Xiao, H. Dong, L. Ma, Z. Liu, and Q. Zhang, "HS-BAS: A hybrid storage system based on band awareness of shingled write disk," in *Proc. IEEE 34th Int. Conf. Comput. Design (ICCD)*, Oct. 2016, pp. 64–71.

[15] C.-L. Wang, D. Wang, Y. Chai, C. Wang, and D. Sun, "Larger, cheaper, but faster: SSD-SWD hybrid storage boosted by a new SMR-oriented cache framework," in *Proc. IEEE Symp. Mass Storage Syst. Technol. (MSST)*, May 2017.

[16] Z.-W. Lu and G. Zhou, "Design and implementation of hybrid shingled recording raid system," in *Proc. IEEE 14th Int. Conf. Pervasive Intell. Comput. (PiCom)*, Aug. 2016, pp. 937–942.

[17] D. Luo, J. Wan, Y. Zhu, N. Zhao, F. Li, and C. Xie, "Design and implementation of a hybrid shingled write disk system," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 1017–1029, Apr. 2016.

[18] V. Tkachenko. (2016). *Flash Cache*. [Online]. Available: https://github.com/facebookarchive/flashcache

[19] EMC Corporation. (2013). *VNX FAST Cache—A Detailed Review*. [Online]. Available: http://www.emc.com/collateral/software/white-papers/h8046-clariion-celerra-unified-fast-cache-wp.pdf

[20] D. Reinsel and J. Rydning, "Breaking the 15K-rpm HDD performance barrier with solid state hybrid drives," IDC, Framingham, MA, USA, White Paper #244250, 2013.

[21] *Dell Storage SC9000 Array Controller, Specification Sheet*, Dell Inc., Round Rock, TX, USA, 2017.

[22] A. Offeri, D. Hartmanii, and B. Porat, "IBM XIV Gen3 storage system model 2810-114 120,000 mailbox resiliency exchange 2010 storage solution," White Paper, 2017.

[23] *Netapp FAS9000 Modular Hybrid Flash System*, NetApp Corp., Sunnyvale, CA, USA, 2017.

[24] *Huawei OceanStor Storage Cases, Specification Sheet*, Huawei Technol. Co., Ltd., Shenzhen, China, 2017.

[25] *Dell EMC Unity Hybrid Storage, Specification Sheet*, Dell Inc., Round Rock, TX, USA, 2017.

[26] L. Lin, Y. Zhu, J. Yue, Z. Cai, and B. Segee, "Hot random off-loading: A hybrid storage system with dynamic data migration," in *Proc. IEEE 19th Annu. Int. Symp. Model., Anal., Simulation Comput. Telecommun. Syst. (MASCOTS)*, Jul. 2011, pp. 318–325.

[27] X. Chen, W. Chen, Z. Lu, P. Long, S. Yang, and Z. Wang, "A duplication-aware SSD-based cache architecture for primary storage in virtualization environment," *IEEE Syst. J.*, vol. 11, no. 4, pp. 2578–2589, Dec. 2017.

[28] Y. Oh, E. Lee, C. Hyun, J. Choi, D. Lee, and S. H. Noh, "Enabling cost-effective f lash based caching with an array of commodity SSDs," in *Proc. 16th Annu. Middleware Conf.*, 2015, pp. 63–74.

[29] X. Meng *et al.*, "HFA: A hint frequency-based approach to enhance the I/O performance of multi-level cache storage systems," in *Proc. 20th IEEE Int. Conf. Parallel Distrib. Syst. (ICPADS)*, 2014, pp. 376–383.

[30] Y. Liu, X. Ge, X. Huang, and D. H. C. Du, "MOLAR: A cost-efficient, high-performance hybrid storage cache," in *Proc. Int. Conf. Cluster Comput. (CLUSTER)*, 2013, pp. 1–5.

[31] J. Tai, B. Sheng, Y. Yao, and N. Mi, "SLA-aware data migration in a shared hybrid storage cluster," *Cluster Comput.*, vol. 18, no. 4, pp. 1581–1593, 2015.

[32] S. Huang, Q. Wei, D. Feng, J. Chen, and C. Chen, "Improving flash-based disk cache with lazy adaptive replacement," *ACM Trans. Storage*, vol. 12, no. 2, 2016, Art. no. 8.

[33] T. Kgil and T. Mudge, "FlashCache: A NAND flash memory file cache for low power Web servers," in *Proc. Int. Conf. Compil., Archit. Synth. Embedded Syst. (CASES)*, 2006, pp. 103–112.

[34] T. Kgil, D. Roberts, and T. Mudge, "Improving NAND flash based disk caches," in *Proc. 35th Int. Symp. Comput. Archit. (ISCA)*, 2008, pp. 327–338.

[35] Y. Oh, J. Choi, D. Lee, and S. H. Noh, "Caching less for better performance: balancing cache size and update cost of flash memory cache in hybrid storage systems," in *Proc. USENIX Conf. File Storage Technol. (FAST)*, 2012, pp. 25:1–25:12.

[36] J. Yang, N. Plasson, G. Gillis, N. Talagala, S. Sundararaman, and R. Wood, "HEC: Improving endurance of high performance flash-based cache devices," in *Proc. 6th Int. Syst. Storage Conf. (SYSTOR)*, 2013, Art. no. 10.

[37] J. Ou, J. Shu, Y. Lu, L. Yi, and W. Wang, "EDM: An endurance-aware data migration scheme for load balancing in SSD storage clusters," in *Proc. IEEE 28th Int. Conf. Parallel Distrib. Process. Symp. (IPDPS)*, May 2014, pp. 787–796.

[38] R. Appuswamy, D. C. van Moolenbroek, and A. S. Tanenbaum, "Cache, cache everywhere, flushing all hits down the sink: On exclusivity in multilevel, hybrid caches," in *Proc. 29th Symp. Mass Storage Syst. Technol. (MSST)*, May 2013, pp. 1–14.

[39] Y. Cheng, W. Chen, Z. Wang, X. Yu, and Y. Xiang, "AMC: An adaptive multi-level cache algorithm in hybrid storage systems," *Concurrency Comput., Pract. Exper.*, vol. 27, no. 16, pp. 4230–4246, 2015.

[40] Y. Chai, Z. Du, X. Qin, and D. A. Bader, "WEC: Improving durability of SSD cache drives by caching write-efficient data," *IEEE Trans. Comput.*, vol. 64, no. 11, pp. 3304–3316, Nov. 2015.

[41] N. Dai, Y. Chai, Y. Liang, and C. Wang, "ETD-cache: An expiration-time driven cache scheme to make SSD-based read cache endurable and cost-efficient," in *Proc. 12th ACM Int. Conf. Comput. Frontiers (CF)*, 2015, Art. no. 26.

[42] F. Ye, J. Chen, X. Fang, J. Li, and D. Feng, "A regional popularity-aware cache replacement algorithm to improve the performance and lifetime of SSD-based disk cache," in *Proc. IEEE Int. Conf. Netw., Archit. Storage (NAS)*, Aug. 2015, pp. 45–53.

[43] H.-P. Chang, S.-Y. Liao, D.-W. Chang, and G.-W. Chen, "Profit data caching and hybrid disk-aware completely fair queuing scheduling algorithms for hybrid disks," *Softw., Pract. Exper.*, vol. 45, no. 9, pp. 1229–1249, 2015.

[44] M. Saxena and M. M. Swift, "Design and prototype of a solid-state cache," *ACM Trans. Storage*, vol. 10, no. 3, 2014, Art. no. 10.

[45] Y. Li, L. Guo, A. Supratak, and Y. Guo, "Enabling performance as a service for a cloud storage system," in *Proc. IEEE 7th Int. Conf. Cloud Comput. (CLOUD)*, Jun./Jul. 2014, pp. 554–561.

[46] Z. Zong, R. Fares, B. Romoser, and J. Wood, "FastStor: improving the performance of a large scale hybrid storage system via caching and prefetching," *Cluster Comput.*, vol. 17, no. 2, pp. 593–604, 2014.

[47] S. Lee, Y. Won, and S. Hong, "Mining-based file caching in a hybrid storage system," *J. Inf. Sci. Eng.*, vol. 30, no. 6, pp. 1733–1754, 2014.

[48] W. Felter, A. Hylick, and J. Carter, "Reliability-aware energy management for hybrid storage systems," in *Proc. IEEE 27th Symp. Mass Storage Syst. Technol. (MSST)*, May 2011, pp. 1–13.
[49] K. Bu, M. Wang, H. Nie, W. Huang, and B. Li, "The optimization of the hierarchical storage system based on the hybrid SSD technology," in *Proc. 2nd Int. Conf. Intell. Syst. Design Eng. Appl. (ISDEA)*, 2012, pp. 1323–1326.
[50] M. Canim, G. A. Mihaila, B. Bhattacharjee, K. A. Ross, and C. A. Lang, "SSD bufferpool extensions for database systems," *Proc. Very Large Data Bases Endowment*, vol. 3, nos. 1–2, pp. 1435–1446, 2010.
[51] 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. (MASCOTS)*, 2007, pp. 402–409.
[52] M. Saxena, M. M. Swift, and Y. Zhang, "FlashTier: A lightweight, consistent and durable storage cache," in *Proc. 7th ACM Eur. Conf. Comput. Syst. (EuroSys)*, 2012, pp. 267–280.
[53] K. R. Krish, B. Wadhwa, M. S. Iqbal, M. M. Rafique, and A. R. Butt, "On efficient hierarchical storage for big data processing," in *Proc. 16th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGrid)*, May 2016, pp. 403–408.
[54] D. Zhao, K. Qiao, and I. Raicu, "Towards cost-effective and high-performance caching middleware for distributed systems," *Int. J. Big Data Intell.*, vol. 3, no. 2, pp. 92–110, 2015.
[55] J. Do, D. Zhang, J. M. Patel, D. J. DeWitt, J. F. Naughton, and A. Halverson, "Turbocharging DBMS buffer pool using SSDs," in *Proc. Int. Conf. Manage. Data (SIGMOD)*, 2011, pp. 1113–1124.
[56] D. Lee, C. Min, and Y. I. Eom, "Effective SSD caching for high-performance home cloud server," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2015, pp. 152–153.
[57] S. H. Baek and K.-W. Park, "A fully persistent and consistent read/write cache using flash-based general SSDs for desktop workloads," *Inf. Syst.*, vol. 58, pp. 24–42, Jun. 2016.
[58] Y. Liu, J. Huang, C. Xie, and Q. Cao, "RAF: A random access first cache management to improve SSD-based disk cache," in *Proc. IEEE 5th Int. Conf. Netw., Archit. Storage (NAS)*, Jul. 2010, pp. 492–500.
[59] STEC Inc. (2012). *EnhanceIO Open Source for Linux*. [Online]. Available: https://github.com/stec-inc/EnhanceIO
[60] D. K. Mridha and L. Bert, "Elastic cache with single parity," U.S. Patent 9 122 629, Sep. 1, 2015.
[61] *HP SmartCache*. (2016). [Online]. Available: https://www.hpe.com/us/en/product-catalog/detail/pip.5364342.html
[62] Y. Liang, Y. Chai, N. Bao, H. Chen, and Y. Liu, "Elastic queue: A universal SSD lifetime extension plug-in for cache replacement algorithms," in *Proc. 9th ACM Int. Syst. Storage Conf. (SYSTOR)*, 2016, pp. 1–11.
[63] G. Yadgar, M. Factor, K. Li, and A. Schuster, "Management of multi-level, multiclient cache hierarchies with application hints," *ACM Trans. Comput. Syst.*, vol. 29, no. 2, p. 5, 2011.
[64] Z. Zhang, Y. Kim, X. Ma, G. Shipman, and Y. Zhou, "Multi-level hybrid cache: Impact and feasibility," Oak Ridge Nat. Lab., Oak Ridge, TN, USA, Tech. Rep., 2012, doi: 10.2172/1035823.
[65] Y. Klonatos, T. Makatos, M. Marazakis, M. D. Flouris, and A. Bilas, "Azor: Using two-level block selection to improve SSD-based I/O caches," in *Proc. 6th IEEE Int. Conf. Netw., Archit. Storage (NAS)*, Jul. 2011, pp. 309–318.
[66] 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. (ICS)*, 2011, pp. 22–32.
[67] Z. Li, "GreenDM: A versatile tiering hybrid drive for the trade-off evaluation of performance, energy, and endurance," Ph.D. dissertation, Dept. Comput. Sci., Stony Brook Univ., Stony Brook, NY, USA, 2014.
[68] 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. IEEE 19th Annu. Int. Symp. Modeling, Anal., Simulation Comput. Telecommun. Syst. (MASCOTS)*, Jul. 2011, pp. 227–236.
[69] A. Raghavan, A. Chandra, and J. B. Weissman, "Tiera: Towards flexible multi-tiered cloud storage instances," in *Proc. 15th Int. Middleware Conf.*, 2014, pp. 1–12.
[70] R. Salkhordeh, H. Asadi, and S. Ebrahimi, "Operating system level data tiering using online workload characterization," *J. Supercomput.*, vol. 71, no. 4, pp. 1534–1562, 2015.
[71] 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 *Proc. Int. Conf. Swarm Intell.*, 2012, pp. 527–534.
[72] J. Xue, F. Yan, A. Riska, and E. Smirni, "Storage workload isolation via tier warming: How models can help," in *Proc. 11th Int. Conf. Auto. Comput. (ICAC)*, 2014, pp. 1–11.
[73] J. Guerra, H. Pucha, J. S. Glider, W. Belluomini, and R. Rangaswami, "Cost effective storage using extent based dynamic tiering," in *Proc. USENIX Conf. File Storage Technol. (FAST)*, 2011, pp. 20–31.
[74] G. Zhang, L. Chiu, and L. Liu, "Adaptive data migration in multi-tiered storage based cloud environment," in *Proc. 3rd Int. Conf. Cloud Comput. (CLOUD)*, 2010, pp. 148–155.
[75] A. Elnably, H. Wang, A. Gulati, and P. J. Varman, "Efficient QoS for multi-tiered storage systems," in *Proc. USENIX Workshop Hot Topics Storage File Syst. (HotStorage)*, 2012, p. 6.
[76] H. Wang and P. Varman, "Balancing fairness and efficiency in tiered storage systems with bottleneck-aware allocation," in *Proc. USENIX Conf. File Storage Technol. (FAST)*, 2014, pp. 229–242.
[77] G. Zhang, L. Chiu, C. Dickey, L. Liu, P. Muench, and S. Seshadri, "Automated lookahead data migration in SSD-enabled multi-tiered storage systems," in *Proc. IEEE 26th Symp. Mass Storage Syst. Technol. (MSST)*, May 2010, pp. 1–6.
[78] X. Zhao, Z. Li, and L. Zeng, "FDTM: Block level data migration policy in tiered storage system," in *Proc. IFIP Int. Conf. Netw. Parallel Comput. (NPC)*, 2010, pp. 76–90.
[79] H. Shi, R. V. Arumugam, C. H. Foh, and K. K. Khaing, "Optimal disk storage allocation for multi-tier storage system," in *Proc. Asia–Pacific Magn. Rec. Conf. (APMRC)*, 2012, pp. 1–7.
[80] X. Wu and A. L. N. Reddy, "Data organization in a hybrid storage system," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, 2012, pp. 583–587.
[81] S. Ma, H. Chen, Y. Shen, H. Lu, B. Wei, and P. He, "Providing hybrid block storage for virtual machines using object-based storage," in *Proc. 20th IEEE Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2014, pp. 150–157.
[82] I. Iliadis, J. Jelitto, Y. Kim, S. Sarafijanovic, and V. Venkatesan, "ExaPlan: Queueing-based data placement and provisioning for large tiered storage systems," in *Proc. IEEE 23rd Int. Symp. Modeling, Anal. Simulation Comput. Telecommun. Syst. (MASCOTS)*, Oct. 2015, pp. 218–227.
[83] 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. 1–4.
[84] X. Wu and A. L. N. Reddy, "Exploiting concurrency to improve latency and throughput in a hybrid storage system," in *Proc. IEEE Int. Symp. Modeling, Anal. Simulation Comput. Telecommun. Syst. (MASCOTS)*, Aug. 2010, pp. 14–23.
[85] D. Park and D. H. C. Du, "Hot data identification for flash-based storage systems using multiple bloom filters," in *Proc. USENIX Conf. File Storage Technol. (FAST)*, 2011, pp. 1–11.
[86] 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. (CIKM)*, 2013, pp. 1631–1636.
[87] S. H. I. Joseph and S. Roy, "Enhancing tiering storage performance," U.S. Patent 8 996 808, Mar. 31, 2015.
[88] D. Montgomery, "Extent migration for tiered storage architecture," U.S. Patent 8 627 004, Jan. 7, 2014.
[89] S. Hayashi and N. Komoda, "Evaluation of volume tiering method and SSD cache method in tiered storage system," in *Proc. 2nd Asian Conf. Inf. Syst. (ACIS)*, 2013, pp. 8–14.
[90] X. Wu and A. L. N. Reddy, "A novel approach to manage a hybrid storage system," *J. Commun.*, vol. 7, no. 7, pp. 473–483, 2012.
[91] 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.
[92] K. Oe and K. Okamura, "A hybrid storage system composed of on-the-fly automated storage tiering (OTF-AST) and caching," in *Proc. 2nd Int. Symp. Comput. Netw. (CANDAR)*, Dec. 2014, pp. 406–411.
[93] K. Oe, T. Nanri, and K. Okamura, "On-the-fly automated storage tiering with caching and both proactive and observational migration," in *Proc. 3rd Int. Symp. Comput. Netw. (CANDAR)*, 2015, pp. 371–377.
[94] M. Abashkin, A. Natanzon, and E. Bachmat, "Integrated caching and tiering according to use and QoS requirements," in *Proc. IEEE 34th Int. Perform. Comput. Commun. Conf. (IPCCC)*, Dec. 2015, pp. 1–8.
[95] Q. Yang and J. Ren, "I-CASH: Intelligently coupled array of SSD and HDD," in *Proc. IEEE 17th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2011, pp. 278–289.

[96] G. Soundararajan, V. Prabhakaran, M. Balakrishnan, and T. Wobber, "Extending SSD lifetimes with disk-based write caches," in *Proc. USENIX Conf. File Storage technol. (FAST)*, 2010, pp. 101–114.

[97] C. Li, D. Feng, Y. Hua, and F. Wang, "Improving RAID performance using an endurable SSD cache," in *Proc. 45th Int. Conf. Parallel Process. (ICPP)*, 2016, pp. 396–405.

[98] 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. IEEE 10th Int. Symp. Parallel Distrib. Process. Appl.*, Jul. 2012, pp. 403–410.

[99] M. K. Aguilera, K. Keeton, A. Merchant, K.-K. Muniswamy-Reddy, and M. Uysal, "Improving recoverability in multi-tier storage systems," in *Proc. 37th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, 2007, pp. 677–686.

[100] B. Mao, H. Jiang, S. Wu, Y. Fu, and L. Tian, "SAR: SSD assisted restore optimization for deduplication-based storage systems in the cloud," in *Proc. 7th Int. Conf. Netw., Archit. Storage (NAS)*, 2012, pp. 328–337.

[101] C. Wu, X. He, Q. Cao, and C. Xie, "Hint-K: An efficient multi-level cache using K-step hints," in *Proc. 39th Int. Conf. Parallel Process.*, 2010, pp. 624–633.

[102] T. M. Wong and J. Wilkes, "My cache or yours? Making storage more exclusive," in *Proc. General Track Annu. Conf. USENIX Annu. Tech. Conf.*, 2002, pp. 161–175.

[103] B. S. Gill, "On multi-level exclusive caching: Offline optimality and why promotions are better than demotions," in *Proc. USENIX Conf. File Storage Technol. (FAST)*, 2008, Art. no. 4.

[104] T. Bisson and S. A. Brandt, "Flushing policies for NVCache enabled hard disks," in *Proc. Mass Storage Syst. Technol. (MSST)*, 2007, pp. 299–304.

[105] L. Shi, J. Li, C. J. Xue, and X. Zhou, "Hybrid nonvolatile disk cache for energy-efficient and high-performance systems," *ACM Trans. Design Autom. Electron. Syst.*, vol. 18, no. 1, 2013, Art. no. 8.

[106] T.-C. Huang and D.-W. Chang, "TridentFS: A hybrid file system for non-volatile RAM, flash memory and magnetic disk," *Softw., Pract. Exper.*, vol. 46, no. 3, pp. 291–318, 2016.

[107] H. Kim, S. Seshadri, C. L. Dickey, and L. Chiu, "Evaluating phase change memory for enterprise storage systems: A study of caching and tiering approaches," *ACM Trans. Storage*, vol. 10, no. 4, pp. 15:1–15:21, 2014.

[108] G. Dhiman, R. Ayoub, and T. Rosing, "PDRAM: A hybrid PRAM and DRAM main memory system," in *Proc. ACM/IEEE 46th Design Autom. Conf. (DAC)*, 2009, pp. 664–669.

[109] (2017). *Cache Replacement Policies—Wikipedia, The Free Encyclopedia*. [Online]. Available: https://en.wikipedia.org/wiki/Cache_replacement_policies

[110] L. Cherkasova, "Improving WWW proxies performance with greedy-dual-size-frequency caching policy," Hewlett-Packard Lab., Palo Alto, CA, USA, Tech. Rep. HPL-98-69 (R.1), 1998.

[111] E. J. O'Neil, P. E. O'Neil, and G. Weikum, "The LRU-K page replacement algorithm for database disk buffering," *ACM SIGMOD Rec.*, vol. 22, no. 2, pp. 297–306, 1993.

[112] S. Jin and A. Bestavros, "Popularity-aware greedy dual-size Web proxy caching algorithms," in *Proc. 20th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2000, pp. 254–261.

[113] S. Rajasekaran, S. Duan, W. Zhang, and T. Wood, "Multi-cache: Dynamic, efficient partitioning for multi-tier caches in consolidated VM environments," in *Proc. IEEE Int. Conf. Cloud Eng. (IC2E)*, 2016, pp. 182–191.

[114] Y. Letian, C. Hao, and Z. Liu, "Solid-state disk caching the top-K hard-disk blocks selected as a function of access frequency and a logarithmic system time," U.S. Patent 8 838 895, Sep. 16, 2014.

[115] A. Schaefer and M. Gries, "Adaptive address mapping with dynamic runtime memory mapping selection," U.S. Patent 9 026 767, Jun. 23, 2015.

[116] Y. Lv, X. Chen, G. Sun, and B. Cui, "A probabilistic data replacement strategy for flash-based hybrid storage system," in *Proc. Asia–Pacific Web Conf. (APWeb)*, 2013, pp. 360–371.

[117] D. Park, Y. J. Nam, B. Debnath, D. H. C. Du, Y. Kim, and Y. Kim, "An on-line hot data identification for Flash-based storage using sampling mechanism," *ACM SIGAPP Appl. Comput. Rev.*, vol. 13, no. 1, pp. 51–64, 2013.

[118] J.-W. Hsieh, L.-P. Chang, and T.-W. Kuo, "Efficient on-line identification of hot data for flash-memory management," in *Proc. ACM Symp. Appl. Comput.*, 2005, pp. 838–842.

[119] X. Zhao, Z. Li, and L. Zeng, "A hierarchical storage strategy based on block-level data valuation," in *Proc. 4th Int. Conf. Netw. Comput. Adv. Inf. Manage. (NCM)*, vol. 1. 2008, pp. 36–41.

[120] M. Alshawabkeh, A. Riska, A. Sahin, and M. Awwad, "Automated storage tiering using Markov chain correlation based clustering," in *Proc. 11th Int. Conf. Mach. Learn. Appl. (ICMLA)*, 2012, pp. 392–397.

[121] X. Meng, L. Zheng, L. Li, and J. Li, "PAM: An efficient power-aware multi-level cache policy to reduce energy consumption of software defined network," in *Proc. 1st Int. Conf. Ind. Netw. Intell. Syst. (INISCom)*, 2015, pp. 18–23.

[122] J. Tai, B. Sheng, Y. Yao, and N. Mi, "Live data migration for reducing SLA violations in multi-tiered storage systems," in *Proc. Int. Conf. Cloud Eng. (IC2E)*, 2014, pp. 361–366.

[123] D. Park, B. Debnath, Y. Nam, D. H. C. Du, Y. Kim, and Y. Kim, "HotDataTrap: A sampling-based hot data identification scheme for flash memory," in *Proc. 27th Annu. ACM Symp. Appl. Comput. (SAC)*, 2012, pp. 1610–1617.

[124] M. He, L. Xing, and G. Li, "A data migration strategy for HSM based on data value," *J. Inf. Comput. Sci.*, vol. 8, no. 2, pp. 312–317, 2011.

[125] X. Zhao, Z. Li, X. Zhang, and L. Zeng, "Block-level data migration in tiered storage system," in *Proc. 2nd Int. Conf. Comput. Netw. Technol. (ICCNT)*, 2010, pp. 181–185.

[126] Y. Cheng, F. Douglis, P. Shilane, G. Wallace, P. Desnoyers, and K. Li, "Erasing Belady's limitations: In search of flash cache offline optimality," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, 2016, pp. 379–392.

[127] W. Burkhard and J. Palmer, "Rotational position optimization (RPO) disk scheduling," Univ. California San Diego, La Jolla, CA, USA, Tech. Rep., 2001. [Online]. Available: http://www.ncstrl.org:8900/ncstrl/servlet/search?formname=detail&id=oai%3Ancstrlh%3Aucsd_cs%3Ancstrl.ucsd_cse%2F%2FCS2001-0679

[128] D. M. Jacobson and J. Wilkes, "Disk scheduling algorithms based rotational position," Hewlett-Packard Lab., Palo Alto, CA, USA, Tech. Rep. HPL-CSP-91-7rev1, 1991.

[129] P. J. Denning, "Effects of scheduling on file memory operations," in *Proc. Apr. 18–20, 1967, Spring Joint Comput. Conf.*, Apr. 1967, pp. 9–21.

[130] Y. Cho and T. Kim, "An efficient scheduling algorithm for NCQ within SSDs," *IEICE Electron. Exp.*, vol. 12, no. 4, p. 20150066, 2015.

[131] B. Jun and D. Shin, "Workload-aware budget compensation scheduling for NVMe solid state drives," in *Proc. Non-Volatile Memory Syst. Appl. Symp. (NVMSA)*, 2015, pp. 1–6.

[132] C. Gao, L. Shi, M. Zhao, C. J. Xue, K. Wu, and E. H. Sha, "Exploiting parallelism in I/O scheduling for access conflict minimization in flash-based solid state drives," in *Proc. 30th Symp. Mass Storage Syst. Technol. (MSST)*, 2014, pp. 1–11.

[133] Z. Yao, I. Papapanagiotou, and R. D. Callaway, "Multi-dimensional scheduling in cloud storage systems," in *Proc. Int. Commun. Conf. (ICC)*, 2015, pp. 395–400.

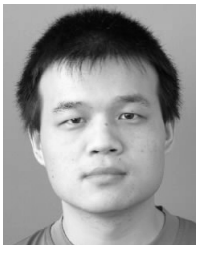
[134] S. Gougeaud, S. Zertal, J.-C. Lafoucriere, and P. Deniel, "A generic and open simulation tool for large multi-tiered hierarchical storage systems," in *Proc. Int. Symp. Perform. Eval. Comput. Telecommun. Syst. (SPECTS)*, 2016, pp. 1–8.

[135] Y. Yamato, "Use case study of HDD-SSD hybrid storage, distributed storage and HDD storage on openstack," in *Proc. 19th Int. Database Eng. Appl. Symp.*, 2015, pp. 228–229.

[136] C. San-Lucas and C. L. Abad, "Towards a fast multi-tier storage system simulator," in *Proc. IEEE Ecuador Tech. Chapters Meeting (ETCM)*, Oct. 2016, pp. 1–5.

[137] *SPC Benchmark 1C (SPC-1) SPC Benchmark 1C/Energy Extension (SPC-1C/E) Official Specification*, Storage Performance Council, 2012.

[138] *OLTP Application I/O and Search Engine I/O*, UMass Trace Repository, 2007.



JUNPENG NIU received the B.E. and M.S. degrees from the School of Computer Engineering, Nanyang Technological University, Singapore, in 2007 and 2009, respectively, where he is currently pursuing the Ph.D. degree with the School of Electrical and Electronic Engineering.

JUN XU (SM'12) received the B.S. degree in applied mathematics from Southeast University, China, in 2001, and the Ph.D. degree in control and automation from Nanyang Technological University, Singapore, in 2006. He was with the Data Storage Institute, Nanyang Technological University and the National University of Singapore. He is currently a Principle Engineer with HGST Western Digital Corporation. He has published around 60 international papers and patents, and one monograph. He has multi-discipline knowledge and solid experiences in complex system design, management, modeling and simulation, data analytics, data center, cloud storage, and IoT. He is a certificated FRM. He was a committee member of several international conferences on control and automation. He is an Editor of the journal *Unmanned Systems*.

LIHUA XIE (F'07) received the B.E. and M.E. degrees in electrical engineering from the Nanjing University of Science and Technology, Nanjing, China, in 1983 and 1986, respectively, and the Ph.D. degree in electrical engineering from the University of Newcastle, Callaghan, NSW, Australia, in 1992.

Since 1992, he has been with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, where he is currently a Professor and the Director of the Delta-NTU Corporate Laboratory for Cyber-Physical Systems. He served as the Head of the Division of Control and Instrumentation from 2011 to 2014. His research interests include robust control and estimation, networked control systems, multiagent networks, localization, and unmanned systems. He is an Elected Member of Board of Governors and the IEEE Control System Society from 2016 to 2018. He is a fellow of IFAC. He is an Editor-in-Chief of *Unmanned Systems* and an Associate Editor of the IEEE TRANSACTIONS ON NETWORK CONTROL SYSTEMS. He has served as an Editor of IET book series in *Control* and an Associate Editor of a number of journals, including the IEEE TRANSACTIONS ON AUTOMATIC CONTROL, *Automatica*, the IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-II.