HiSMRfs: a High Performance File System for Shingled Storage Array

Chao JIN, Wei-Ya XI✉, Zhi-Yong CHING, Feng HUO, Chun-Teck LIM
Data Storage Institute, Agency of Science, Technology and Research, Singapore
✉Corresponding Author: Xi_Weiya@dsi.a-star.edu.sg
{Jin_Chao, Ching_Zhi_Yong, Huo_Feng, Lim_Chun_Teck}@dsi.a-star.edu.sg

***Abstract*—HiSMRfs, a file system with standard POSIX interface suitable for Shingled Magnetic Recording (SMR) drives, has been designed and developed. HiSMRfs can manage raw SMR drives and support random writes without remapping layer implemented inside SMR drives. To achieve high performance, HiSMRfs separates data and metadata storage, and manages them differently. Metadata is managed using in-memory tree structures and stored in a high performance random write area such as in a SSD. Data writing is done through sequential appending style and store in a SMR drive. HiSMRfs includes a file/object-based RAID module for SMR/HDD arrays. The RAID module computes parity for individual files/objects and guarantees that data and parity writing are 100% in sequential and in full stripe. HiSMRfs is also suitable for a hybrid storage system with conventional HDDs and SSDs. Two prototype systems with HiSMRfs have been developed. The performance has been tested and compared with SMRfs and Flashcache. The experimental tests show that HiSMRfs performs 25% better than SMRfs, and 11% better than Flashcache system.**

I. Introduction

A. Technologies for High Density Storage

Traditional magnetic hard drives are fast approaching their limits in capacity growth [1]. Various technologies such as Shingled Magnetic Recording (SMR), Heat-Assisted Magnetic Recording (HAMR), Bit-Patterned Media Recording (BPMR), and Helium filled drives, are being developed to increase data density. SMR overlaps, or shingles, adjacent tracks to increase disk areal density. In Gibsons Shingling Geometry Model [2], a SMR disk density can be more than two times higher than a conventional HDD. Both HAMR and BPMR have their own innovative methods of packing data bits even more closely together. However, these technologies face the challenge of fabrication/assembly. SMR is based on Perpendicular Magnetic Recording (PMR) technology and does not require much change in their fabrication/assembly. Helium filled drives can reduce air flow-induced mechanical vibrations which is helpful to increase track density from the mechanical point of view. SMR drives can be used together with either PMR or HAMR or BPMR. It can also be used in a helium filled drive. Three disk drive manufacturers, Seagate, Western Digital and Toshiba, have SMR disk drives in their product lines.

B. SMR drives - Problem and Approaches

A major problem with SMR is that write updates may erase previously written data. Therefore in a SMR disk, data writing is restricted to sequential writes through appending [1].

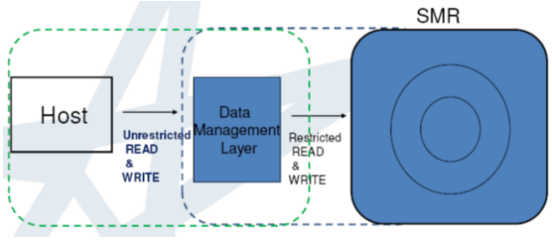

Fig. 1. Data Management Layer for SMR

To address this problem, a data management layer has to be designed and developed to redirect all update-in-place requests to different locations so that a host can send unrestricted read/write commands to a SMR storage device. The data management layer can be implemented in the disk drive itself, or in the host, or in both locations as shown in Figure 1. If the data management layer is implemented inside the disk drive, this approach is referred to as drive-managed SMR. If the data management layer is implemented in the host, it is referred to as host-managed SMR. If part of the data management layer functions is implemented inside the host and the rest inside the disk, this approach is called cooperatively-managed SMR [4].

C. Related Work on SMR

Researchers and developers from both research institutes and industry have proposed various approaches to solve the problem so that SMR drives can be used for wider applications rather than just for backup/archival (Read Many Write Once). Gibson and Ganger presented research directions and possible solutions for SMR drive [3]. Amer et.al. [5], [6], [18] presented various possible data layouts and analysis for SMR drives. Pitchumani et.al. [7] described a method of emulating a SMR drive using a HDD. Lin et.al. [8] studied garbage collection performance and proposed a hot data based garbage collection scheme for SMR. Tan et.al. [14] designed and developed simulation software for shingled disk simulation.

There have also been several significant works done for drive-managed SMR solutions. Cassuto et.al. [9] proposed an indirection system for disk drive firmware. Kasiraj et.al. [17] presented a strategy to separate the shingled zone from the non-shingled zone which can coexist in a single drive. Chang et.al. [11] proposed a data layout strategy called floating guard band for SMR. Feldman and Gibson presented a cooperatively-

978-1-4799-5671-5/14/$31.00

managed SMR proposal for mostly sequential writing applications [4].

Although a drive-managed SMR solution can reduce the dependency to the file systems/host system, a host-managed SMR will provide more flexibility and performance benefits. HiSMRfs adopts a host-managed approach. Besides HiSMRfs, other reported works have also adopted the host-managed approach. One is the log-structure file system for SMR disks. This idea was proposed by New et.al. [10] but so far no reported system has claimed to have implemented this. Another is a hybrid architecture proposed by Gibson et.al. using SMR drives together with SSDs for big data applications [3]. The source codes of this work, SMRfs, have been released recently [23].

In the log-structure file system, all writings and modifications are performed through sequential appending making it suitable for SMR drives. However, the log-structure file system does not separate the file metadata from the file data. These are mixed and kept together. File metadata updates are random and happen frequently and, in a conventional disk drive system, is usually the bottleneck of the system. In a SMR storage, frequent metadata updates can severely impact SMR storage performance. To date, the authors are not aware of any implementation of the log-structure file system in a SMR system/prototype.

Both SMRfs and HiSMRfs separate metadata from data storage and management. Metadata is stored in a high performance and non-sequential writing storage space to improve the file systems performance. The major difference between HiSMRfs and SMRfs is that HiSMRfs implements an in-memory metadata tree structure and hash tables are designed to speed up metadata lookup of a file within a directory, while SMRfs stores the metadata in the extended attribute of a symbol link file in a conventional file system (e.g. Ext4 or Btrfs). In addition, SMRfs is for fast prototyping with tmpfs employed as a simple implementation of a read/write cache. When a file is opened, the entire file needs to be copied to tmpfs, a part of system memory, in order to perform any read/write operation to the file. On the other hand, HiSMRfs implements a complete file request queuing and scheduling management module which can largely increase the performance and decrease the system memory required.

The rest of this paper is arranged as follows. Section 2 presents details of the design and development of HiSMRfs. Section 3 describes implementation of the prototype and presents and discusses the results of performance tests. A summary and suggestions for future work are contained in the last section of this paper.

II. Design and Development of HiSMRfs

A. SMR Drive Data Layout

There are two possible data layouts for a SMR drive. In one, the disk media contains only multiple SMR bands/zones and each band/zone are separated by guard bands. In this design, all tracks on the disks are shingled. Compared to a SMR drive which contains some non-shingled tracks, this layout can provide a much higher amount of storage capacity. However, all data writing within the shingled bands/zones has to be done sequentially. In the other layout of a SMR drive, the disk media contains one or more conventional magnetic recording (CMR) bands/zones and multiple SMR bands/zones. In the CMR bands/zones, tracks are not overlapped so that update-in-place can be performed. The CMR bands/zones in SMR drives are also called random write zones where random writes/updates can be performed.

Compared to the previous layout with all shingled tracks, this layout consumes much more disk media space to store the same amount of data. This is because the write head in a SMR drive is normally much bigger (could be 3 times bigger) than the size of the write head in a conventional HDD. Thus the track-to-track distance of the CMR bands/zones in a SMR drive has to be much wider than that in a conventional disk. This means, for the same disk area occupied by a CMR in a SMR drive, the useful data storage capacity provided by the CMR zone is only one third that provided in a conventional disk drive. Having CMR bands/zones coexisting with SMR bands/zones in a SMR drive is therefore a very significant disadvantage. Because of this, it is thus highly unlikely that a SMR drive with both CMR zones and SMR zones will become available commercially. Hereafter in this paper, a SMR drive refers to one which has only shingled tracks on the disk media. The HiSMRfs can be used for this type of SMR drives.

B. File System Metadata and Data Accesses

In a storage system, there are normally two types of data. One is the file metadata and the other is the actual file data. File metadata are data that provides useful information about the actual data stored in the file. File metadata are small in size but need to be accessed/updated frequently. The actual file data, on the other hand, are usually much larger and do not need to be updated as often as their metadata.

To have an estimate of how often file metadata are accessed compared to file data, an experiment was conducted on a personal computer with a hard disk drive attached and running under Linux. Filebench was used to generate the workload for the disk drive which was formatted using the Ext4 file system. A predefined number of files, ranging from 50K to 800K, was generated and stored in the disk drive. A file server application then generated a workload to access the files stored in the disk. Blktrace was used to count number of file data accesses to the disk. The number of metadata accesses was computed from the total number of disk accesses less the number of file data accesses. The ratio of the number of metadata accesses to the number of file data reads was then computed. This ratio varies with the total number of files stored on the disk and the relationship is shown in Figure 2.

It can be seen from Figure 2 that as the number of files stored in the disk increases, a larger number of metadata IOs is needed to access a single file data. With 800K files stored, to read a data file requires, on average, three metadata accesses. Normally in enterprise storage, the number of file stored is much bigger than 800k. For a storage with 1TB capacity, if each file size is 1MB, the total number of files stored is about 1 million. File metadata requests are usually random, requiring random accesses to disk storage, and small in size. Especially for a disk with a large number of stored files, this is likely to become the bottleneck limiting the file

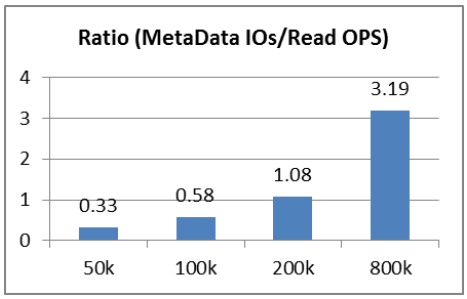

Fig. 2. Ratio of number of metadata IOs to number of file read IOs

system performance. To improve storage system performance, it is therefore more effective and critical to improve the performance of file metadata accesses rather than those of file data accesses [20].

C. Data Writes/Updates of a SMR Drive

There are two main ways that a SMR drive performs data writes/updates. One is update-in-place through read-modify-write, and the other is to write at the end through appending.

In update-in-place, once an update request is received, the whole data band/zone in which the to-be-updated data file is stored is first read into memory. Updating of the data file within the data band/zone is then performed in memory and once this is done, the whole updated data band/zone is written back to disk storage into the same original location.

In writing at the end through appending, when an update request is received, the new data is written directly at the end of a data band/zone through appending. The old data is then invalidated and the space released will be claimed through a subsequent garbage collection process.

The two ways of data writing/updating result in different performance of a SMR drive. Le et.al. [18] conducted tests to compare the numbers of block movements in SMR drives for the two writing approaches to the number of block movements in conventional hard disk drives. The results showed that, the scheme with SMR write at the end through appending has about one third of the number of block movements when compared to that of a conventional hard disk drive. The fewer the number of block movements involved, the better the disk performance will be. Therefore, HiSMRfs adopted the writing at the end through appending approach when writing file data to a SMR drive.

D. HiSMRfs

HiSMRfs has been designed, developed and implemented at the Linux user space. Figure 3 shows the block diagram of the architecture of HiSMRfs. HiSMRfs provides a standard POSIX interface to user applications. To achieve better performance, the metadata and data are totally separated in the HiSMRfs and managed differently by two modules, the Metadata Management module and the File Data Management module. Through these two modules, the metadata and file data are written to the unshingled and shingled partitions respectively and separately and directly from user space through raw-device read/write interfaces.

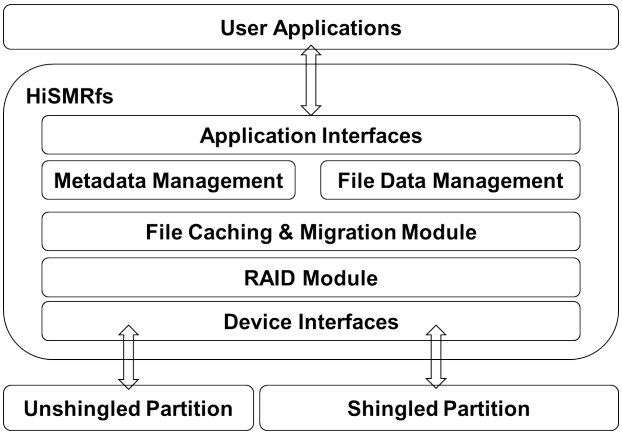

Fig. 3. Architecture of HiSMRfs

The File Caching and Placement module identifies hot file data to be cached in the unshingled partition when extra storage space is available and where data storage/retrieval is much faster there. The minimum capacity required for the unshingled partition is to store all the metadata of the HiSMRfs. This is about 1∼2% of the storage capacity occupied by the file data in the shingled partition. When there is extra storage space available at the unshingled partition, for example when the number of files has not reached the capacity of the disk storage system, the File Caching and Placement module will identify hot file data to be cached there. A simple algorithm for file data caching, based on file data size and access frequency, has been implemented in the HiSMRfs.

HiSMRfs can also be used to optimize the performance of storage systems with both high and low performance zones, with high performance zone corresponding to the unshingled partition, and low performance zone corresponding to the shingled partition. In a conventional HDD, the high performance zone can be located at the outermost diameter of the disk media as data transfer rates is highest there. For hybrid drives, the high performance zone can be the consolidated storage space of the NVM/flash embedded in these drives and the low performance zone can be the consolidated disk media storage space. In a hybrid storage system, the high performance zone can also be a SSD and the low performance zone can be the SMR drives.

1) Metadata Management: Figure 4 illustrates the in-memory metadata structure of the HiSMRfs. The file directory hierarchy is represented by a tree structure. Starting from the root directory, each file or directory is allocated a node in the tree structure. Each node stores the metadata information of the file or directory. The node for each file also records its file extent list to indicate where the file content is stored in the device. The file and directory nodes are connected through the children, parent, and neighbors pointers.

In addition, hash tables are used to speed up the metadata lookup for a file within a directory. As file creation, modification and deletion operations cause changes to the metadata structure, these metadata operations are recorded with timestamps in a log file stored in the unshingled partition. Periodically, the whole metadata structure is synchronized into the metadata file as persistent checkpoints in the unshingled

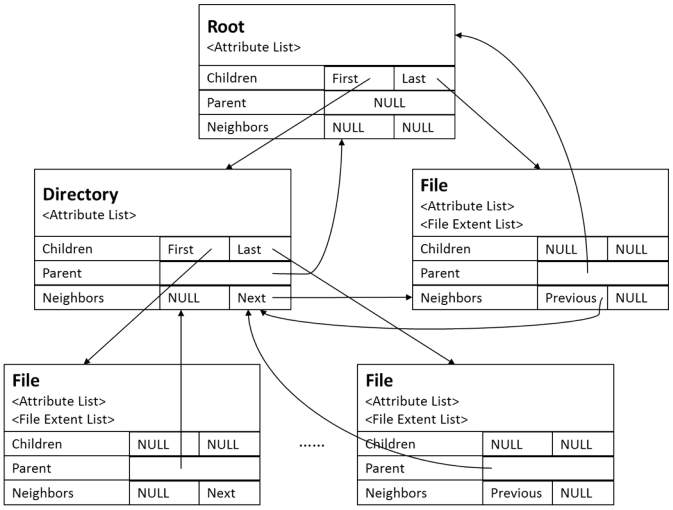

Fig. 4. Metadata Structure

partition.

2) File Data Management: File data, including newly written data and modifications to existing data, is sequentially appended at the end of each band/zone in the low performance zone built in the SMR drives. The File Data Management system implements four major modules, including File Data Allocation module, Garbage Collection module, Request Queue Scheduling module and Band/Zone Layout module. The File Data Allocation module determines where the data will be written. File Request Queuing and Scheduling module arranges file read/write requests into queues and efficiently schedule them to increase performance and decrease memory usage of the file system when large files are accessed.

File deletion and modification will cause invalid data blocks in the data log, and these invalid blocks need to be reclaimed to free storage space. The Garbage Collection module is responsible to reclaim released space. Two different approaches have been designed for garbage collection in HiSMRfs. One is the file-based approach, in which all the files are copied sequentially to a new place, and the old data blocks, including the invalid blocks, are freed. The other is the band-based approach. In this approach, several candidate bands are first selected to be reclaimed and the valid blocks in these bands are moved to a new band freeing all these candidate bands.

The Band/Zone Layout module emulates a SMR data layout and its related information and passed the information to the File Data Allocation module and the Garbage Collection module for them to issue data access requests to access data.

3) RAID Module: Conventional RAID systems implement data redundancy at the block level. While block-level RAID provides independency to the above file systems, it lacks the file system information and shows several disadvantages when compared with file-level RAID. In HiSMRfs RAID functions are implemented at the file system level. HiSMRfs is thus capable of working on an array of SMR storage devices, and providing, in addition, good fault tolerance. With the information from the file system, HiSMRfs further optimizes the RAID reconstruction processes in the event of fault occurrences or device failures.

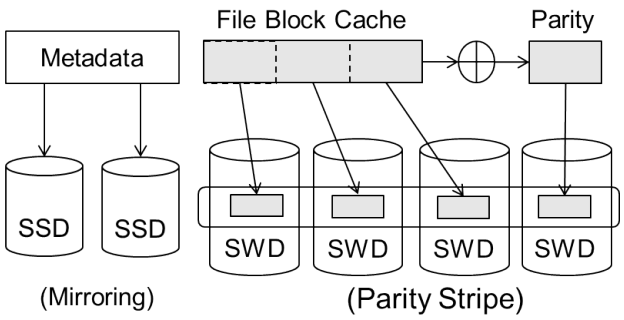

Fig. 5. Structure of RAID in HiSMRfs

Figure 5 shows how the RAID module is implemented in HiSMRfs. HiSMRfs implements the RAID function at the file level. Each file block in HiSMRfs is chopped into multiple equal-sized sub-blocks, and their XOR sum computed as the parity sub-block. While this is just an example for HiSMRfs to tolerate a single disk failure, it is clear that this can be easily extended in HiSMRfs to tolerate multiple disk failures by using erasure codes. Each of the sub-blocks is written to one of the SMR devices in the array correspondingly. In other words, HiSMRfs organizes each file block as an independent parity stripe, and writes this to the disk array through a full-stripe write. The file blocks are written to the disk array in using appending mode, ensuring that writes to each of the SMR/HDD device also follows the appending style. On the other hand, metadata in the unshingled partition (e.g., SSD) is protected by mirroring.

With the RAID function implemented at the file level in HiSMRfs, optimization of the data reconstruction process can be achieved when disk failures occur. By traversing the metadata tree, only the failed blocks which are allocated by the file blocks need to be reconstructed, skipping the other invalid or free blocks.

III. Prototype and Performance Test

The HiSMRfs can be used for either SMR drives or conventional HDDs. There is no SMR drive available in the market yet, therefore two prototype systems are developed using HDDs. In Prototype I, the storage consists of one SSD and one HDD. In the other Prototype II, the storage consists of one SSD and five HDDs. Extensive tests were conducted on both prototypes to evaluate their performance. These tests showed that HiSMRfs can perform 25% better than SMRfs system and 11% better than EXT4 with Flashcache [22].

A. Prototype I

Table III-A shows hardware and software configuration of storage array Prototype I. The storage Prototype I consists of one SSD and one HDD. The server is installed with Linux and filebench.

The HiSMRfs Prototype I has been tested under workload of Creatfile and Varmail. These results are used to compare with published test results of SMRfs [23]. Figure 6 shows the performance comparison of HiSMRfs and SMRfs under Createfiles workload. Figure 7 is the performance comparison

TABLE I. PROTOTYPE I CONFIGURATION

Item	Specification
Server	Intel(R) Xeon(TM) CPU quad core @3.00GHz with 4GB memory
SSD	ZUES Z16IFE3B 73GB
HDD	Seagate Barracuda ST380013AS 7200rpm
Linux	Kernel 2.6.35 Fuse version 2.9.2
Filebench	Version 1.4.9

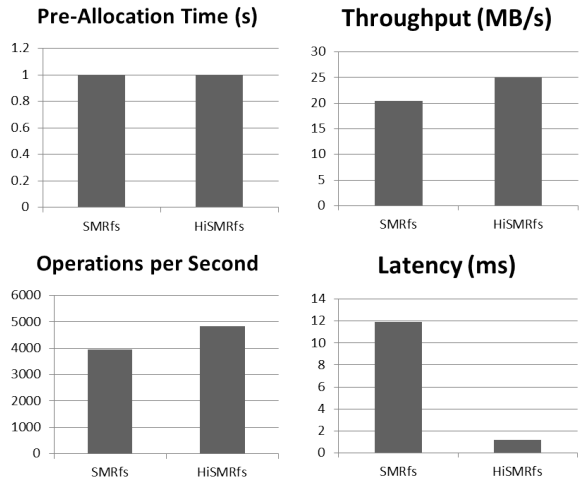

Fig. 6. Performance Comparison of HiSMRfs and SMRfs under Createfile Workload

of the two systems under Varmail workload. It can be seen that for both workloads, the pre-allocation time used for both systems are about the same. For Creatfile workload, the HiSMRfs is about 20% better than the SMRfs in terms of IOPS and bandwidth; while for Varmail workload, the HiSMRfs is about 30% better than the SMRfs in terms of OPS and bandwidth. For both workloads, the HiSMRfs demonstrates significant shorter latency than the SMRfs. From the tests conducted, it can be seen that HiSMRfs shows more advantages when the system is under varmail application workload. Varmail

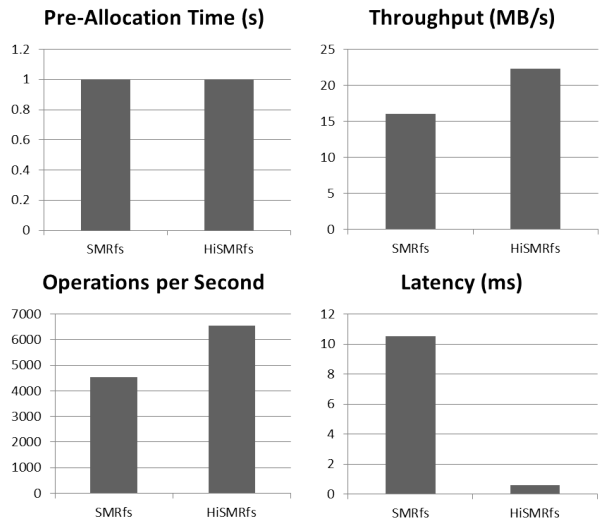

Fig. 7. Performance Comparison of HiSMRfs and SMRfs under Varmail Workload

TABLE II. PROTOTYPE II CONFIGURATION

Item	Specification
Server	Intel(R) Xeon(TM) CPU E5645 @2.4GHz with 2GB memory
SSD x1	MTRON SSD MOBI 3500 SATA 16GB
HDD x5	Maxtor Maxline Plus II 250GB 7200rpm
RAID Level	RAID5
Linux	Kernel 3.10.0 Fuse version 2.9.2
Filebench	Version 1.4.9
Flashcache	Version 3.0

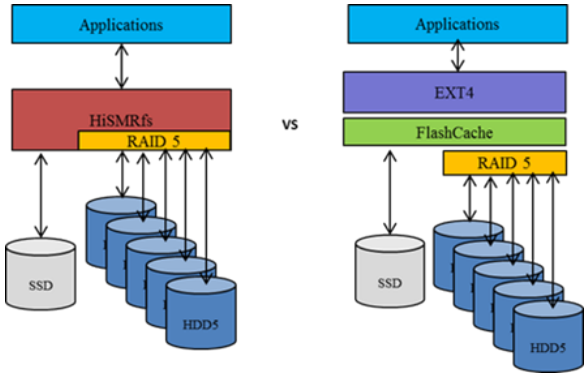

Fig. 8. Topology of two systems built on Prototype II

system normally consists of a lot of small sized files and also a lot of metadata of the files. HiSMRfs can speed up metadata accesses, and therefore the overall system performance can be improved significantly.

B. Prototype II

Table III-B shows hardware and software configuration of storage array Prototype II. The storage of the Prototype II consists of five Maxtor HDDs and one SSD. The server is installed with Linux and Filebench.

Two different systems have been built on Prototype II to test and compare their performances. One is the HiSMRfs and the other is EXT4 with Flashcache. Figure 8 shows the topology of the two systems. On the left side of the Figure 8 is the HiSMRfs system and on the right side is the EXT4 with Flashcache system. The HiSMRfs manages the SSD as high performance zone and five HDDs as low performance zone. The HiSMRfs RAID module configures the HDDs as RAID 5. Flashcache uses the SSD as the cache and five HDDs configured RAID5 as its storage. Flashcache virtualizes underlying storage including both SSD and HDDs and presents them to EXT4 as single storage volume.

Prototype II was first setup as HiSMRfs system and conducted the tests. Once the tests are done, the Prototype II is then setup as EXT4 with Flashcache system on the same hardware and then tested under the same workload as the one used for the HiSMRfs system.

In actual system, high performance storage capacity such as SSD is limited and it is normally fully utilized, in other words, millions of files need to be generated in order to fully utilize up the SSD capacity. To reduce files to be generated which can take long time, the amount of SSD capacity used for the test is reduced to 1GB for both systems under test so that less number of files needs to be generated.

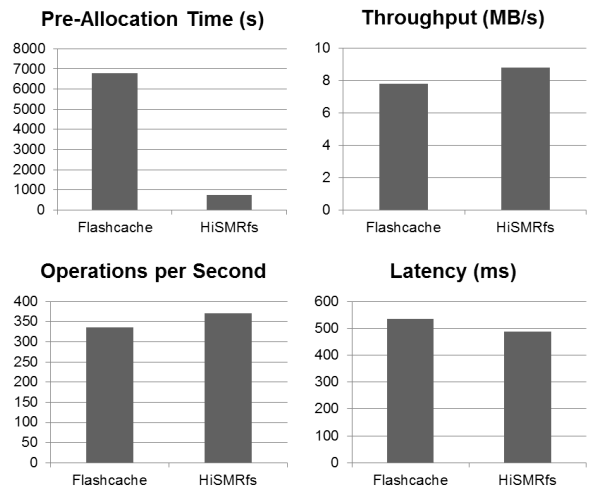

Fig. 9. Performance comparisons between HiSMRfs and EXT4 with Flashcache under file server workload

File server application workload consisting of 200K files is generated by Filebench to conduct the test. Figure 9 shows performance results of the two systems under tests. It can be seen that HiSMRfs can perform better than the Flashcache system on data allocation, OPS, throughput and latency. The pre-allocation is about 9 times shorter than the Flashcache system. HiSMRfs can performance about 11% better than Flashcache system in terms of OPS, throughput and latency.

In the test, Flashcache used nearly 100% of the 1GB SSD space, while HiSMRfs just used the SSD to store metadata, which is about 50MB. Therefore, HiSMRfs can provide better performance than Flashcache even with much less SSD resource usage.

IV. Summary

HiSMRfs is a high performance file system with standard POSIX interface. It is designed to efficiently manage a storage system consisting of two different storage media with different performance characters. It is suitable to manage a storage consisting of SMR drive array with all shingled bands. The HiSMRfs supports random writes and can provide high performances so that SMR drives can be used for wider applications such as those require random writes and high performances.

Two prototype systems implemented with HiSMRfs have been developed. The first prototype is implemented in single-node mode and compared with SMRfs. The experimental results show that HiSMRfs can perform 25% faster than SMRfs system. The second prototype is implemented in disk-array mode and compared with Flashcache. The performance test results show that HiSMRfs, even with much less SSD resource usage, can perform 11% better than Flashcache.

References

[1] Y. Shiroishi, K. Fukuda, I. Tagawa, S. Takenoiri, H. Tanaka, and N. Yoshikawa. *Future options for HDD storage*. IEEE Transactions on Magnetics, vol. 45, no. 10, Oct. 2009.

[2] G. Gibson and G. Ganger. *Principles of Operation for Shingled Disk Devices*. Carnegie Mellon University Parallel Data Lab Technical Report, CMU-PDL-11-107, April 2011.

[3] G. Gibson and M. Polte. *Directions for shingled-write and two dimensional magnetic recording system architectures: synergies with solid state disks*. Carnegie Mellon University Parallel Data Lab Technical Report, CMU-PDL-09-104, May 2009.

[4] T. Feldman and G. Gibson. *Shingled magnetic recording areal density increase requires new data management*. USENIX issue, Vol. 38, No. 3, June 2013.

[5] A. Amer, D. D. E. Long, E. L. Miller, J.-F. Paris, and S. J. T. Schwarz. *Design issues for a shingled write disk system*. In Proceedings of IEEE Symposium on Mass Storage Systems and Technologies (MSST), May 2010.

[6] A. Amer, J. Holliday, D. D. E. Long, E. L. Miller, J.-F. Paris, T. Schwarz. *Data Management and Layout for Shingled Magnetic Recording*. IEEE Transactions on Magnetics, vol. 47, no. 10, October 2011.

[7] Rekha Pitchumani, Andy Hospodor, Ahmed Amer, Yangwook Kang, Ethan L. Miller and Darrell D. E. Long. *Emulating a shingled writing disk*. In Proceedings of the 20th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), August 2012.

[8] Chung-I Lin, Dongchul Park, Weiping He and David H.C. Du. *H-SWD: Incorporating Hot Data Identi?cation into Shingled Write Disks*. In Proceedings of the 20th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), August 2012.

[9] Y. Cassuto, M. Sanvido, C. Guyot, D. Hall, and Z. Bandic. *Indirection Systems for Shingled-Recording Disk Drives*. In Proceedings of IEEE Symposium on Mass Storage Systems and Technologies (MSST), May 2010.

[10] Richard New, Mason Williams. *Log-structured file system for disk drives with shingled writing*. US patent 20050071537.

[11] Dar-Der Chang, Ken Hong, Byeung Jun Lee, Xin Guo. *Floating Guard Band for Shingled Magnetic Recording*. US Patent 20110304935.

[12] C. Jin, W. Y. Xi, K. L. Yong and Z. Y. Ching. *Data storage system, method of writing to storage in the data storage system, hard disk and method of forming the hard disk*. US patent.

[13] W. Y. Xi, S. Tan, K. L. Yong, C. T. Lim, Z. Y. Ching and C. Jin. *Architecture design and method to store data in hybrid shingled writing disk*. US patent.

[14] S. Tan, W. Y. Xi, Z. Y. Ching, C. Jin, and C. T. Lim. *simulation for a shingled magnetic recording disk*. IEEE Transaction on Magnetics. March 2013.

[15] W. Y. Xi, S. Tan, Z. Y. Ching, T. C. Low, X. J. Wu, E. Toh, C. Jin, Y. Jia. *SS_sim - Network and Storage Simulation Part 1: Performance Simulation*. International Journal of Advancement in Computing Technology (ISSN: 2005-8093), to be published.

[16] W. Y. Xi, W. K. For, D. H. Wang, R. Kanagavalu, W. K. Koh. *OSDsim: a Simulation and Design Platform of an Object-based Storage Device*. In Proceeding of 14th NASA Goddard, 23rd IEEE Conference on Mass Storage Systems and Technologies (MSST), May, 2006.

[17] P. Kasiraj, R. New, J. de Souza, and M. Williams. *System and method for writing data to dedicated bands of a hard disk drive*. US patent 7490212.

[18] Q. M. Le, K. SathyanarayanaRaju, A. Amer, and J. Holliday. *Workload Impact on Shingled Write Disks: All-Writes Can Be Alright*. In Proceedings of IEEE Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), July 2011.

[19] W. Y. Xi, S. Tan, Z. Y. Ching, T. C. Low, X. J. Wu, E. Toh, C. Jin, Y. Jia. *SS_sim - Network and Storage Simulation Part 2: Power Consumption Simulation*. International Journal of Advancement in Computing Technology (ISSN: 2005-8093), to be published.

[20] Jaegeuk Kim. *f2fs: introduce flash-friendly file system*. http://http://lwn.net/Articles/518718/, Samsung.

[21] Mike Yan. *Open computer project, cold storage hardware*. http://www.opencompute.org/projects/storage/, Facebook.

[22] Mohan Srinivasan. *Flashcache: A Write Back Block Cache for Linux*. https://github.com/facebook/flashcache, Facebook.

[23] CMU SMR Wiki. https://wiki.pdl.cmu.edu/smr/webhome.