Managing Storage Space in a Flash and Disk Hybrid Storage System

Xiaojian Wu, A. L. Narasimha Reddy
Department of Electrical and Computer Engineering, Texas A&M University
College Station, Texas 77843
Email: (tristan.woo@, reddy@ece.)tamu.edu

***Abstract*—This paper considers the problem of efficiently managing storage space in a hybrid storage system employing flash and disk drives. The flash and disk drives exhibit different performance characteristics of read and write behavior. We propose a technique for balancing the workload properties across flash and disk drives in such a hybrid storage system. The presented approach automatically and transparently manages migration of data blocks among flash and disk drives based on their access patterns. This paper presents the design and an evaluation of the proposed approach on a Linux testbed through realistic experiments.**

I. Introduction

Traditionally, storage systems and file systems have been designed considering the characteristics of magnetic disks e.g., seek and rotational delays. When both flash and magnetic disk drives are employed in a single hybrid storage system, a number of these policies may need to be revisited.

Flash based storage devices exhibit different characteristics than magnetic disk drives. For example, writes to flash devices can take longer than magnetic disk drives while reads can finish faster. Flash drives typically have more uniform performance (especially for reads) depending on file size, where magnetic disks typically perform better with larger file sizes. The write performance of flash drives can experience much larger differences in peak to average completion times due to block remapping done in the Flash Translation Layer (FTL) and necessary erase cycles to free up garbage blocks. These differences in characteristics need to be taken into account in hybrid storage systems in efficiently managing and utilizing the available storage space.

Various methods have been proposed to compensate and exploit diversity in device characteristics. Most of these approaches deal with devices of different speeds, storing frequently or recently used data on the faster devices. However, the flash and disk storage devices have asymmetric read/write access characteristics, based on request sizes and whether the requests are reads or writes. This asymmetry makes this problem of managing the different devices challenging and interesting in a flash+disk hybrid system. In this paper, we treat the flash and disk devices as peers and not as two levels in a storage hierarchy.

This work is supported by NSF grants 0702012 and 0621410.

In a traditional storage hierarchy, hot data is moved to the faster (smaller) device and cold data is moved to the larger (slower) device. In our approach, the cold data is still moved to the larger device. However, the hot data may be stored in either device because of the performance asymmetry.

We propose a measurement-driven approach to migration in this paper. We observe the access characteristics of individual blocks and consider migrating individual blocks to another device whose characteristics may better match the block access patterns. Our approach can easily adapt to different devices and changing workloads or access patterns.

The remainder of this paper is organized as follows. Section II gives details of the related work. Section III describes the the proposed approach to managing the space in a hybrid system. Section IV presents our prototype implementation and its evaluation. Section V presents the results of the evaluation. Section VI concludes the paper.

II. Related Work

Flash storage has received much attention recently. Studies have been done on block management and buffer management [11], [12], [13]. Recently, employment of flash in a database workload is considered [10]. This work considered static measurement of device performance in deciding on migration as opposed to our method of employing dynamically measured response times. File system based approaches for flash and non-volatile storage can be found, for example in [15], [16]. We consider a device level approach here and we manage storage across the devices at a block or chunk level instead of at the file level.

Data migration is considered previously in networked devices [4], in large data centers [3], [6]. Data (file) migration has been studied extensively earlier in systems where disks and tapes are used [2], [5], [7]. HP's AutoRAID system considered data migration between a mirrored device and a RAID device [1]. Our work is inspired by much of this earlier work.

III. Measurement-driven Migration

Our general approach is to pool the storage space across flash and disk drives and make it appear like a single larger device to the file system and other layers above. We manage the space across the different devices underneath, transparent to the file system.

	Process 1			Process 2		
	Transactions /sec	Read Throughput	Write Throughput	Transactions /sec	Read Throughput	Write Throughput
Transcend-Hybrid (Striping)	52	211.46	49.22	50	204.70	47.65
Transcend-Hybrid (Migration)	58	237.18	55.21	57	229.68	53.46
MemoRight-Hybrid (Striping)	126	509.17	118.52	125	502.98	117.08
MemoRight-Hybrid (Migration)	137	553.17	128.76	134	538.76	125.41

TABLE I
PERFORMANCE OF DIFFERENT SYSTEMS WITH POSTMARK WORKLOAD. THROUGHPUT IS IN KBYTES/SECOND.

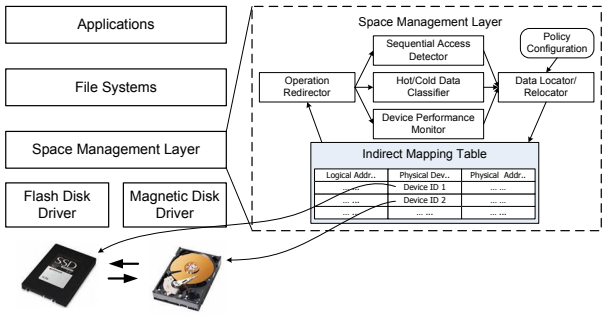


Fig. 1. Architecture of space management layer.

In order to allow blocks to be flexibly assigned to different devices, an indirection map, containing mappings of logical to physical addresses, needs to be maintained. Every read and write request is processed after consulting the indirection map and determining the actual physical location of data. Similar structures have been used by others [1], [4].

We keep track of access behavior of a block by maintaining two counters, one measuring the read and the other write accesses. In our implementation, we maintain this information in memory through a hash table. We employed two bytes for keeping track of read/write frequency separately per chunk (64Kbytes or higher). This translates into an overhead of 0.003% or lower or about 32KB per 1GB of storage.

We take a general approach to managing the device characteristics. Instead of characterizing the devices statically, we keep track of device performance dynamically. Every time a request is served by the device, we keep track of the request response time at that device. We maintain both read and write performance separately since the read, write characteristics can be substantially different as observed earlier. We maintain an exponential average of the device performance by computing average response time = 0.99 * previous average + 0.01 * current sample. Such an approach is used extensively in networking, in measuring round trip times and queue lengths. For each device i, we keep track of the read r_i and write w_i response times.

Given a block j's read/write access history through its access counters R_j and W_j and the device response times, we use the following methodology to determine if a block should be moved to a new location. The current average cost of accessing this block j in its current device i, C_{ji}= $(R_j * r_i + W_j * w_i)/(R_j + W_j)$. The cost of accessing a block with similar access patterns at another device k, C_{jk} (computed similarly using the response times of device k) are compared. If $C_{ji} > (1+\delta) * C_{jk}$, we will consider this block for migration, where $0 < \delta$ is a configurable parameter.

In order to control the rate of migration, we employ a token scheme. The tokens are generated at a predetermined rate. Migration is considered only when a token becomes available.

Migration is carried out in blocks or chunks of 64KB or larger. Larger block size increases migration costs, reduces the size of the indirection map, can benefit from spatial locality or similarity of access patterns.

IV. IMPLEMENTATION

We developed a Linux kernel driver that implements several policies for migration and managing space across flash and disk drives in our system. As shown in Fig. 1, the space management layer sits below the file systems and above the device drivers. We implemented several policies for detecting access patterns of blocks. The sequential access detector identifies if blocks are being sequentially accessed by tracking the average size of sequential access to each block. The device performance monitor keeps track of the read/write request response times at different devices. The hot/cold data classifier determines if a block should be considered hot. The indirection map directs the file system level block addresses to the current location of those blocks on the underlying flash and disk devices. The indirection map is maintained on the hard disk, a memory copy of it allows faster operations.

In the experimental environment, the NFS server was a commodity PC system equipped with an Intel Pentium Dual Core 3.2GHz processor, 1GB of main memory. The magnetic disk used in the experiments was one 7200 RPM, 250G SAMSUNG SATA disk (SP2504C), the flash disk drives are a 16GB Transcend SSD (TS16GSSD25S-S), and a 32GB MemoRight GT drive, which were connected to Adaptec SCSI Card 29160 through a SATA-SCSI converter(ADSALVD160). All the NFS clients are simulated by one load generator in the environment. The operating system on the NFS server was Fedora 7 with a 2.6.21 kernel, and the file system used was the Ext2 file system. The hybrid storage system is connected to the server and a number of other PCs are used to access the server as clients.

V. EVALUATION

A. Workload

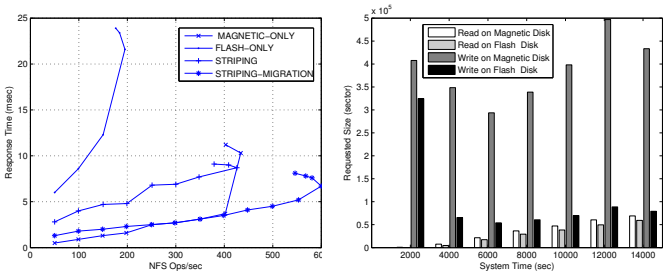
We used multiple workloads for our study. The first workload, SPECsfs benchmark, represents file system workloads [8]. The second workload, Postmark benchmark, represents

typical access patterns in an email server [14]. We also use IOzone [9] benchmark to create controlled workloads at the storage system in order to study the impact of read/write request distributions on the performance of the hybrid system.

We expect these three workloads generated by SPECsfs, IOzone and Postmark to provide insights into the hybrid system performance in typical file system workloads and in other workloads with different read/write request distributions.

B. Benefit From Migration

In the first experiment, we evaluated the benefit from the measurement-driven migration. We compared the performance of the following four policies: FLASH-ONLY, data is allocated on the flash disk only; MAGNETIC-ONLY, data is allocated on the magnetic disk only; STRIPING, data is striped on both disks and STRIPING-MIGRATION, data is striped on and migrated across both disks.



(a) (b)

Fig. 2. (a) Benefit from measurement-driven migration. (b) Request distribution in measurement-driven migration.

The results are shown in Fig. 2(a) for Transcend flash drive. Since there are more write requests in the workload than read requests, and write performance of the flash disk is much worse than that of the magnetic disk, the response times for FLASH-ONLY are longer than those for MAGNETIC-ONLY.

As can be seen from Fig. 2(a), the measurement-based migration has the throughput saturation point at 600 NFS operations/sec, that is much better than 426 NFS operations/sec in the STRIPING policy, and 434 NFS operations/sec in the MAGNETIC-ONLY policy. This improvement benefits from the data redistribution which matches the read/write characteristics of blocks to the device performance.

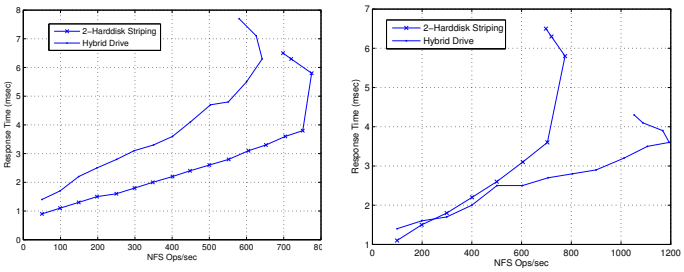
Fig. 2(b) shows the request distribution in different system periods. This request pattern at the end of the simulation shows that our approach is succeeding in redistributing write-intensive blocks to the magnetic disk even though the initial allocation of blocks distributes these blocks equally on the two devices.

This experiment shows that the measurement-driven migration can effectively redistribute the data to the right devices and help decrease the response time while improving the throughput saturation point of the system.

The results with Postmark benchmark experiment are shown in Table I. In our experiment we employed two simultaneous processes to generate sufficient concurrency, each process creating 40,000 files between 500 bytes and 10kB and then performing 400,000 transactions. We ran our experiments across both the hybrid storage systems with Transcend and MemoRight flash drives. It is observed that migration improved the transaction rate, read/write throughputs in both the hybrid storage systems, by about 10%.

C. Comparison with 2-harddisk Device

In this experiment, we compared the performances of the hybrid storage system and a 2-hard disk striping storage system (data is striped on two hard disks and no migration is employed).



(a) (b)

Fig. 3. Hybrid drive vs. 2-harddisk striping device on SPECsfs 3.0. (a) Using Transcend 16G drive. (b) Using MemoRight 32G drive.

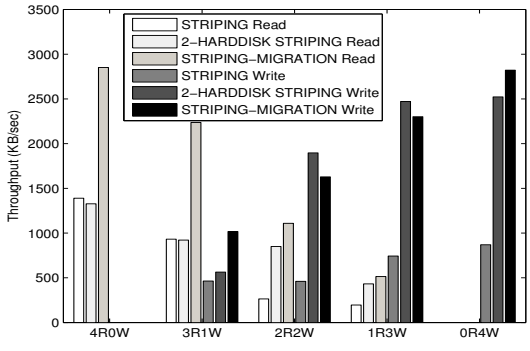


Fig. 4. Hybrid drive vs. 2-harddisk striping device on IOzone.

First, we used SPECsfs 3.0 to generate the workload. As shown in Fig. 3, the MemoRight-based hybrid system outperforms the 2-harddisk striping storage system, while the Transcend-based does not.

We used IOzone to generate workloads with different read/write ratio to find out what kind of workloads are more suitable for the hybrid storage system. In this experiment, we employed four processes to generate workload at the storage system. By varying the number of processes doing reads vs. writes, we could create workloads that 100% writes, to 75%, 50%, 25% or 0% write workloads (0R4W, 1R3W, 2R2W, 3R1W and 4R0W). We employed a Zipf distribution for data accesses in each process and bypassed the cache to maintain a controlled workload at the storage system.

As we can see from Fig. 4, the performance of Transcend-based hybrid drive with STRIPING policy is not as good as that of the 2-harddisk system, especially the writing performance. However, with migration employed, the performance of the hybrid drive achieved significant improvement, even

	Process 1			Process 2		
	Transactions /sec	Read Throughput	Write Throughput	Transactions /sec	Read Throughput	Write Throughput
Transcend-Hybrid (Striping)	17	3.19M	434.88K	17	3.19M	434.69K
Transcend-Hybrid (Migration-1)	18	3.35M	449.43K	17	3.33M	449.75K
Transcend-Hybrid (Migration-2)	19	3.47M	461.86K	18	3.45M	460.23K
MemoRight-Hybrid (Striping)	25	5.39M	610.72K	25	5.33M	602.92K
MemoRight-Hybrid (Migration-1)	26	5.69M	644.48K	26	5.67M	642.35K
MemoRight-Hybrid (Migration-2)	33	5.91M	740.05K	30	6.38M	722.17K
2-Harddisk	24	5.15M	583.10K	24	5.13M	580.32K

TABLE II
PERFORMANCE OF DIFFERENT SYSTEMS WITH POSTMARK WORKLOAD. THROUGHPUT IS IN BYTES/SECOND.

surpassed the 2-harddisk system. The results show that the hybrid drive with migration can get higher performance improvement when the ratio of read/write requests is higher, even with the slower Transcend flash drive. When the ratio was 1:3 (in workload 1R3W), the performances of 2-harddisk system and hybrid drive with STRIPING-MIGRATION policy are almost the same.

These results indicate that read/write characteristics of the workload have a critical impact on the hybrid system. With migration, the hybrid system's performance can be improved greatly and made to offer a performance improvement over a 2-disk system over much wider workload characteristics than it would have been possible.

D. Impacts of Request Size

We conducted an experiment with Postmark benchmark to study the impact of request size. In this experiment, we employed two simultaneous Postmark processes. The file size was varied from a low of 500 bytes to 500K bytes, with each process generating and accessing 4,000 files. Each process conducts 80,000 total transactions in this experiment. The results of these experiment are shown in Table II. We employed two migration policies: Migration-1, only read/write characteristics were considered as earlier and Migration-2, request size was considered as detected by our sequentiality detector module. In the second policy, if the observed request size is less than one migration block (64KB in this experiment), we allowed the block to be migrated based on the read/write request patterns of that block, otherwise we allowed this data to exploit the gain that can be had from striping data across both the devices. If the block is accessed as part of a request larger than 64KB, it will not be migrated.

The results of these two policies are shown in Table II. It is observed that migration policy based on read/write patterns improved the performance over striping. When we considered the request sizes and the read/write access patterns (Migration-2), the performance is observed to be higher. These experiments show that both read/write and request size patterns can be exploited to improve performance.

Memoright-based hybrid storage system provided better performance than a 2 hard disk system, but Transcend-based hybrid system could not match the performance of the 2 hard disk system. The results in this section indicate that the newer Memoright flash drive provides superior performance and exploiting such devices in various scenarios will remain interesting.

VI. CONCLUSION

In this paper, we studied a hybrid storage system employing both flash and disk drives. We have proposed a measurement-driven migration strategy for managing storage space in such a system in order to exploit the performance asymmetry of these devices. Our approach extracts the read/write access patterns and request size patterns of different blocks and matches them with the read/write advantages of different devices. We showed that the proposed approach is effective, based on realistic experiments on a Linux testbed, employing three different benchmarks. The results indicate that the proposed measurement-driven migration can improve the performance of the system significantly, up to 50% in some cases.

REFERENCES

[1] J. Wilkes, R. A. Golding, C. Staelin, T. Sullivan. The HP AutoRAID Hierarchical Storage System. *ACM Trans. Comput. Syst.* 1996.

[2] B. Gavish and O. Sheng. Dynamic File Migration in Distributed Computer Systems. *Commun. ACM*, pages: 177-189, 1990.

[3] L. Yin, S. Uttamchandani and R. Katz. SmartMig: Risk-modulated Proactive Data Migration for Maximizing Storage System Utility. In *Proc. of IEEE MSST* (2006).

[4] S. Kang and A. L. N. Reddy. User-centric data migration in networked storage devices. Proc. of IPDPS, Apr. 2008.

[5] M. Lubeck et al. An Overview of a Large-Scale Data Migration. In *Proc. of IEEE MSST* (2003).

[6] C. Lu, G. A. Alvarez and J. Wilkes Aqueduct: online data migration with performance guarantees In *Proc. of FAST* (2002).

[7] A. J. Smith. Long Term File Migration: Development and Evaluation of Algorithms. *Communications of ACM* 24(8): 521-532 (1981).

[8] Standard Performance Evaluation Corporation. SPEC SFS97_R1 V3.0 *http://www.spec.org/osg/sfs97r1.*

[9] IOZONE file system benchmark. http://www.iozone.org/, accssessed 07/2008.

[10] I. Koltsidas and S. Viglas. Flashing up the storage layer. *Proc. of VLDB*, Aug. 2008.

[11] T. Kgil, D. Roberts and T. Mudge. Improving NAND Flash Based Disk Caches. *Proc. of ACM Int. Symp. Computer Architecture*, June 2008.

[12] H. Kim and S. Ahn. BPLRU: A buffer management scheme for improving random writes in flash storage. *USENIX FAST Conf.*, 2008.

[13] H. Kim and S. Lee. An effective flash memory manager for reliable flash memory space management. *IEICE Trans. on Info. Sys.*, June 2002.

[14] Jeffrey Katcher. Postmark: A New File System Benchmark. *http://www.netapp.com/tech library/3022.html.*

[15] J. Garrison, and A. L. Narasimha Reddy Umbrella File System: Storage management across heterogeneous devices *ACM Trans. on Storage*, 2009.

[16] M.Wu and W. Zwaenepoel. envy: A non-volatile, main storage system. *Proc. of ASPLOS*, 1994.