2015-3-17

High Performance Storage Systems Based on SSD

Topic Related: Hard disk, SSD, RAID

Zhongkai Fan[zhongkai@gwmail.gwu.edu]

GWID: G36899901

High Performance Storage Systems Based on SSD

Abstract

Nowadays SSD (Solid State Drive) has become a common component of computers, especially laptops, this leads the possibility of using in large-scale storage systems. However compared with traditional HDD (Hard Disk Drive), SSD does have much shorter lifetime than HDD. RAID (Redundant Arrays of Inexpensive Disks) is a typical and popular scheme to build large-scale storage systems, to enhance its performance, we propose a storage system consisting of SSDs and HDDs, which uses SSDs to respond to I/O requests, and uses HDDs to backup. This scheme guarantees both performance and reliability reasonably. To make them work effectively together, a nonvolatile memory is used for bridging. To realize the goal, we reconfigure the workloads and HDD. Then multiple HDDs are arranged to supply higher bandwidth. To evaluate this scheme, we first demonstrate this scheme is feasible, next we compare this scheme with other solutions and show the results in the end.

Recently, the technology about flash memory has taken a giant leap. In addition, as the memory storage density increases, the price of per MB decreases, the SSD (Solid State Drive) has drawn wide attention. Compared with hard disks, SSD has lower I/O latency, wider bandwidth, therefore it has been widely deployed in high performance storage systems. However, each storage unit in SSD has a writing limit. For the SSD of SLC (Single Level Cell), manufacturers claim the limit is 100, 000, as for the MLC (Multiple Level Cell), the more times you write to SSD, the lower the reliability it becomes. So many customers cast doubt on the reliability of SSD, especially the enterprise.

As a traditional method to improve reliability, the RAID (Redundant Arrays of Inexpensive Disk) has been widely deployed in the systems based on hard disks. It's not compatible with systems based on SSD for the following reasons:

1. RAID need additional space to store checking message, it's much more expensive using SSD, this would increase the expense of the whole system extraordinarily.
2. To store the checking message, the lifespan of the system could be reduced. For instance, RAID5 makes equivalent amount of checking message while writing data, therefore the lifespan will be reduced by half.
3. Traditional RAID would scatter the I/O requests among disks in it. While using SSD, this could make the lifespan of disks reduce evenly, if one of these SSDs dies because of the limit has been reached, the other disks die in the meantime, the loss of data could happen as a result because multiple disks breakdown.

Based on the analysis above, we think it's hard to construct RAID if only use SSD. Then we come up with a mixed structure using hard disk to support backup function, which the hard disks have a perfect backup of data on SSDs. This system could avoid the defects in a system completely consisting of SSDs. And the advantages are obvious, firstly the total expense won't increase too much as the hard disks are inexpensive. Secondly the lifetime of whole system won't be reduced since hard disks don't have any writing limits. Last but not least, the possibility of multiple disk error in this system is low.

Since the hard disks have higher I/O latency and narrower compared with SSDs, the crux of the matter is how to backup data from SSDs to hard disks efficiently. We think this method is feasible because:

1. The hard disks are only used for backup, which means they only need to respond to writing requests, this will produce lower working load.
2. Compared with hard disks, the writing performance of SSDs is similar. The writing latency of flash memory is high and the garbage collecting of SSD will effect writing performance severely.
3. The hard disk has much higher performance while writing in sequence.

This paper contributes to the following four fields:

1. We implement a mixed RAID structure, which utilizes hard disks to backup SSDs, this reliable storage system is both inexpensive and has high performance.
2. We use I/O reorganization strategy to make the I/O operations sequential in order to improve the writing performance.
3. To avoid sudden writing requests on single unit, we use multiple hard disk to backup, which makes sure the sudden writing data on the SSD could be efficiently

scattered to these hard disks.

4. This method is proved to be feasible by experimenting on prototype system.

1 Background

Section 1.1 first introduces relevant technology, and reveals the mentioned technology couldn’t give consideration to both performance and cost. Section 1.2 mainly talks about the feasibility of this method.

1.1 RAID based on SSD

This section walks through several RAID systems based on SSD, including RAID1, RAID5 and RAID6. For the convenience of analysis, we define Sr as reading latency of SSDs, Sw as writing latency of SSDs, Hrw as reading/writing latency of hard disks because they are almost equivalent for hard disks. Ls indicates the lifespan of each unit in SSDs, Lh indicates the lifespan of hard disks. Let Cs be the cost of SSDs per MB, Ch be the cost of hard disks per MB.

RAID1 based on SSD uses one SSD to back up the other one. When writing request arrives, two SSDs respond at the same time, therefore the writing performance and lifespan are the same as single SSD, the cost doubles though. Since these two SSDs could respond to I/O requests, the reading performance doubles the single SSD. In general, the writing latency is Sw, the lifespan is Ls, and the cost per MB is 2Cs.

The process of RAID5 responding to the write request is as follows: first it reads the user data and checking information before the update, and then it calculates a new checking information. Finally the new user data and checking information are written back to the array. The process involves reading and writing operations twice, but these reading/writing operations are concurrent. Therefore, the writing latency of the RAID5 based entirely on SSD is Sr + Sw. The cost and lifetime of such structure depend on the number of SSDs in the array. Assuming the array consists of N data disks and one checking disk, the price per MB is (N + 1)Cs/N, the average lifespan of each memory cell is (N+1)Ls/2N.

RAID5 and RAID6 resemble each other in principle, except RAID6 uses more checking information disks (2 parity disk) to provide greater reliability , and accordingly , the writing latency of the RAID6 based entirely on SSD is Sr + Sw, the price per MB is (N + 2)Cs/N, the average lifespan of each memory cell is (N+2)Ls/3N.

This paper presents a hybrid structure using disk to back up SSD, so the cost per MB is Cs + Ch, due to lower writing performance of hard disks, the writing latency of entire array depends on hard disks, is Hrw. Due to the limited life of the SSDs, the average lifespan of each memory cell depends on SSD, which is Ls.

Table 1 compares the various structures mentioned above. From Table 1, the hybrid structure this paper presents has the longest life expectancy Ls. Because price per MB of the SSD is 10 times or less , the hybrid structure cost (Cs + Ch) is the lowest (if RAID5 containing more than 10 data disks, the cost will be lower, but the typical RAID5 does not contain so many data disks). The final measurement is the writing performance, the hybrid structure proposed has high write latency, and the main contribution of this paper is how to reduce the write latency. In various structures mentioned above, the read request by SSD response, and therefore have the same read latency, SSD-based RAID1 could read concurrently, and provide twice the read bandwidth.

Table 1 The Comparison of Different Solutions in Term of Write Latency, Cost and Lifespan

Solutions	Write Latency	Cost	Lifespan
RAID1	S_w	$2C_s$	L_s
RAID5	S_r+S_w	$(N+1)C_s/N$	$(N+1)L_s/2N$
RAID6	S_r+S_w	$(N+2)C_s/N$	$(N+2)L_s/3N$
SSD+Disk	H_{rw}	C_s+C_h	L_s

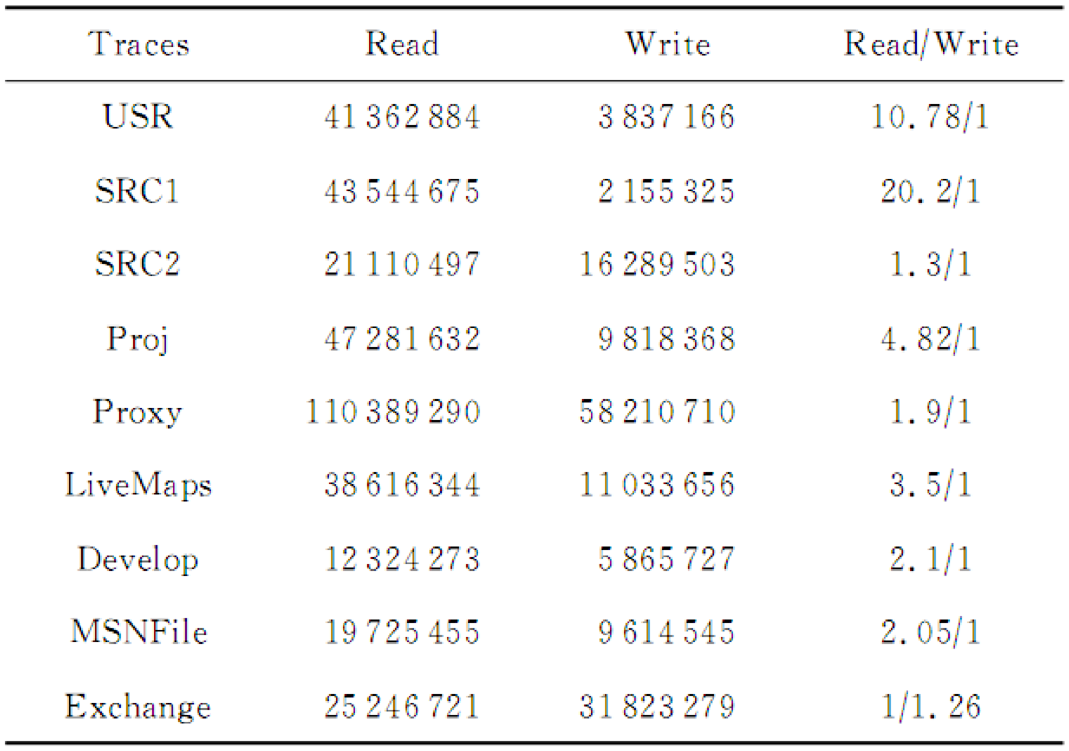
1.2 Feasibility Analysis of Hybrid Backup Structure

From Section 1.1 that disk-based backup and hybrid machines System has the advantage in life expectancy and cost. The key point of building this system is that hard disks can whether backup data timely on SSDs. Next, from both the latency and bandwidth these two aspects to answer this question.

The latency of hard disks is obviously higher than SSDs. Therefore, it's hard to write data on SSDs to hard disks simultaneously. Accordingly, we use nonvolatile memory to bridge the gap between the latency of them. Once the data arrived in nonvolatile memory, we treat it as backed up, the subsequent work could be done in background. As the latency of main memory is lower than SSD, the data could be synchronized to main memory.

Another problem is that whether the hard disk could provide sufficient bandwidth to transfer data in main memory to hard disk. SSD has better performance though, it has to respond to all requests, while the hard disk used for backing only needs to respond to writing requests. Further analysis on I/O factors under typical load is presented in Table 2, the result shows that the reading requests are much frequent than writing requests for most application load. So the working load on SSD is heavier than hard disks, the hard disk does not need to supply with equivalent performance as SSD to back data up.

Table 2 The Characteristic of Traces



Traces	Read	Write	Read/Write
USR	41 362 884	3 837 166	10. 78/1
SRC1	43 544 675	2 155 325	20. 2/1
SRC2	21 110 497	16 289 503	1. 3/1
Proj	47 281 632	9 818 368	4. 82/1
Proxy	110 389 290	58 210 710	1. 9/1
LiveMaps	38 616 344	11 033 656	3. 5/1
Develop	12 324 273	5 865 727	2. 1/1
MSNFile	19 725 455	9 614 545	2. 05/1
Exchange	25 246 721	31 823 279	1/1. 26

In addition, disk sequential writing performance is better, if a large number of writing operations is sequential writing, the disk can provide enough bandwidth. Assuming one writing request contains 61 pages, disk latency for execution of the request is 4 ms, the average write latency of each page is 62.5 us, if the page size is 4KB, equivalent to bandwidth of 64 MBps. In Section 2, we will introduce I/O reorganization strategy which increases the length of sequential I / O, thereby increases the disk reading and writing bandwidth.

In summary, using memory as the bridge between SSD and hard disk, and ensure that sequential writing, the hard disk can provide real time backup services for SSD.

2 I / O Reorganization Strategy

This paper presents one mixed backup mechanism, the crux of this mechanism is to guarantee sequential reads and writes. We have proposed one reorganization strategy to make it work, which increases the sequence of internal operation of disk. Before the detailed discussion, first we will introduce the overall architecture of the proposed system.

2.1 Hybrid Backup Mechanism Architecture

First we will use a basic model to describe the storage structure proposed in this paper: To use a hard disk to backup an SSD, the architecture is shown in Figure 1. Reading requests issued only are responded by the host side SSD, Writing requests are responded by the backup module and SSD at the same time responding. The backup module contains nonvolatile memory, which bridges hard disk and SSD to fill the latency gap. In the internal disk, we propose a reorganization strategy to improve operation sequence, so that the data can be backed up to disk in a timely manner.

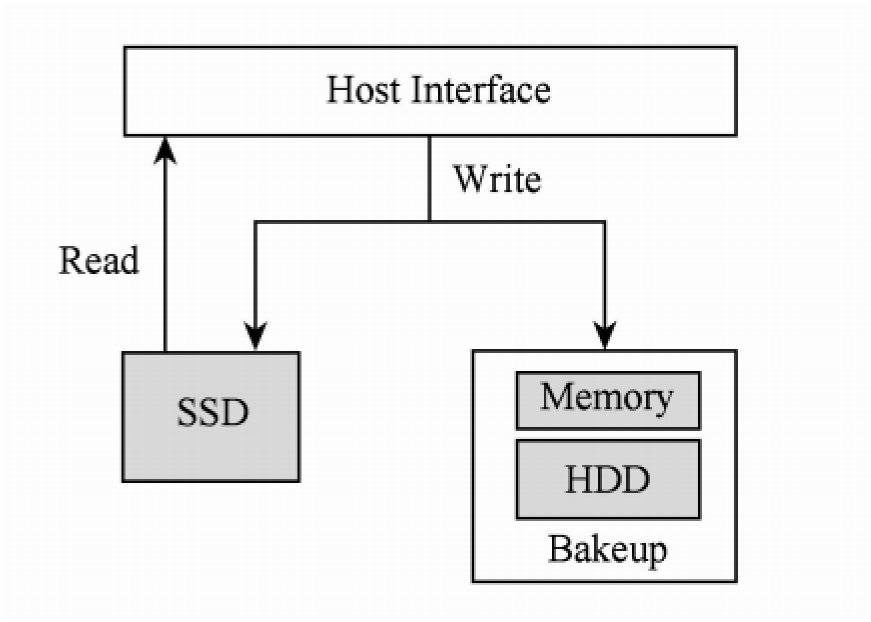
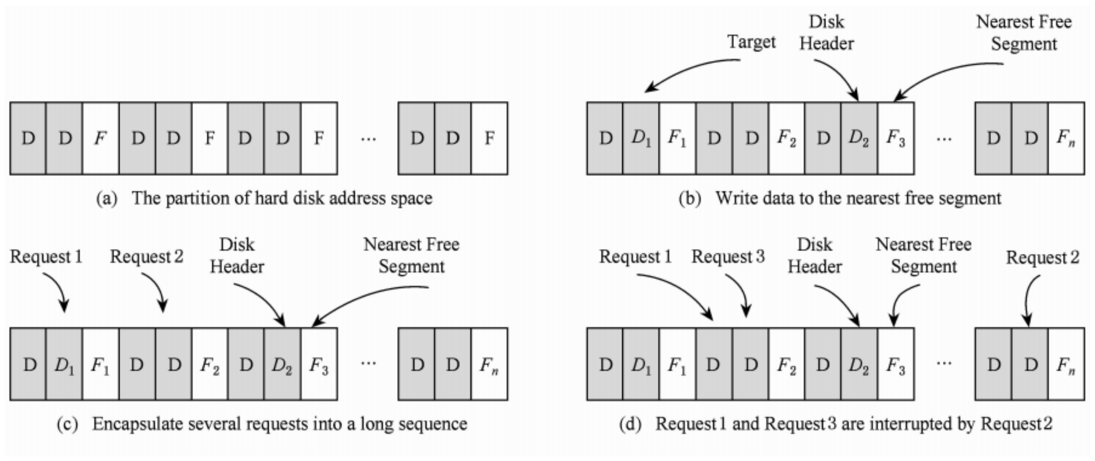


Figure 1 The Architecture of the Backup System

2.2 I/O Reorganization Strategy

The scheduling mechanism within a single hard disk is shown in Figure 2. In Figure 2(a), the disk linear address space is divided into several segments, which are divided into two categories, the label D is called data segment, the label F means free segment. The data segment and free segment are arranged alternately in linear space. As Figure 2 (a) presents, each of the two data segments are before one free segment (In an actual system, there exists a free segment behind dozens of data segments) .SSD data mainly is backed up on the data segment, the free segment is used to temporarily store data. Address division mechanism above has two features.



D D F D D F D D F ··· D D F
(a) The partition of hard disk address space
Target
Disk Header
Nearest Free Segment
D D_1 F_1 D D F_2 D D_2 F_3 ··· D D F_n
(b) Write data to the nearest free segment
Request 1
Request 2
Disk Header
Nearest Free Segment
(c) Encapsulate several requests into a long sequence
Request 1
Request 3
Disk Header
Nearest Free Segment
Request 2
(d) Request1 and Request3 are interrupted by Request2

Figure 2 Reorganization of I/O Requests within the Hard Disk

1. The data segment and free segment are scattered evenly in the addressing space of hard disks, the data segments are used for backup. Therefore, the addressing space of data segments corresponds to the addressing space of SSD, which could be calculated directly without any mapping mechanism.
2. In the address space division strategy mentioned above, the hard disk has a free segment nearby arbitrarily. When a writing request arrives, regardless of magnetic head in the linear address space where there is a free segment nearby, data can be written in a timely manner to the free segment, so that any writing requests could be completed sequentially without reschedule the magnetic head. In Figure 2 (b) we introduce this sequential strategy. As shown in Figure 2 (b), a writing request should be written D1, but the head is now at D2, in order to avoid the head rescheduling, data is not written directly in D1. Instead, we find the nearest free segment Fa, and then write the data. So that the data which should have been written to data segment D1 is now written into the free section Fa. We need to maintain a table to record this correspondence table. The free segments are only used to temporary save user data, and when the disk is idle user data is transferred

to the data segment.

The instances used above indicate the I/O reorganization strategy includes 3 phases:

1. Write data to the closest free segment when the writing requests arrive.
2. Read data from the free segment to nonvolatile memory.
3. Write data to the corresponding data segment.

All I/O operations involved are sequential.

2.3 I/O Sequence Analysis

The I/O reorganization strategy in hard disks is mainly implemented in two ways: The first one is to reduce head rescheduling, the second one is to combine multiple short requests into one long request.

1. **Write data to the closest free segment when the writing requests arrive.**

   When the writing request arrive, the data is written to the nearest free segment to avoid head rescheduling.

   The client always send multiple writing requests in a short time, so it’s effective to combine these writing request into one longer writing request, then write it in to one free segment, which reduces writing operation effectively and make writing more sequential. As Figure 2(c) shows, the client sends two writing requests Request 1 and Request 2, which set two different data segments as destination, while the magnetic head is at D3. This strategy combines these two requests and write data to free segment Fa (the closest free segment). Since every segment has large capacity (Typically this is set to 2MB), multiple writing requests could be combined into one long writing request which heads for the same free segment.

2. **Read data from the free segment to nonvolatile memory.**

   In Step 1 multiple requests are merged into one request, which is written in the same free segment, in Step 2, it only needs one read operation to read all the data, which reduces the number of reading operations.

   In fact, in Step 2, the adjacent free segments are walked through at the same time, which can further reduce the probability of head rescheduling, as shown in Figure 2(c), three adjacent free segments are read simultaneously out, the head rescheduling can be avoided since these idle segments sequentially locate in the disk space.

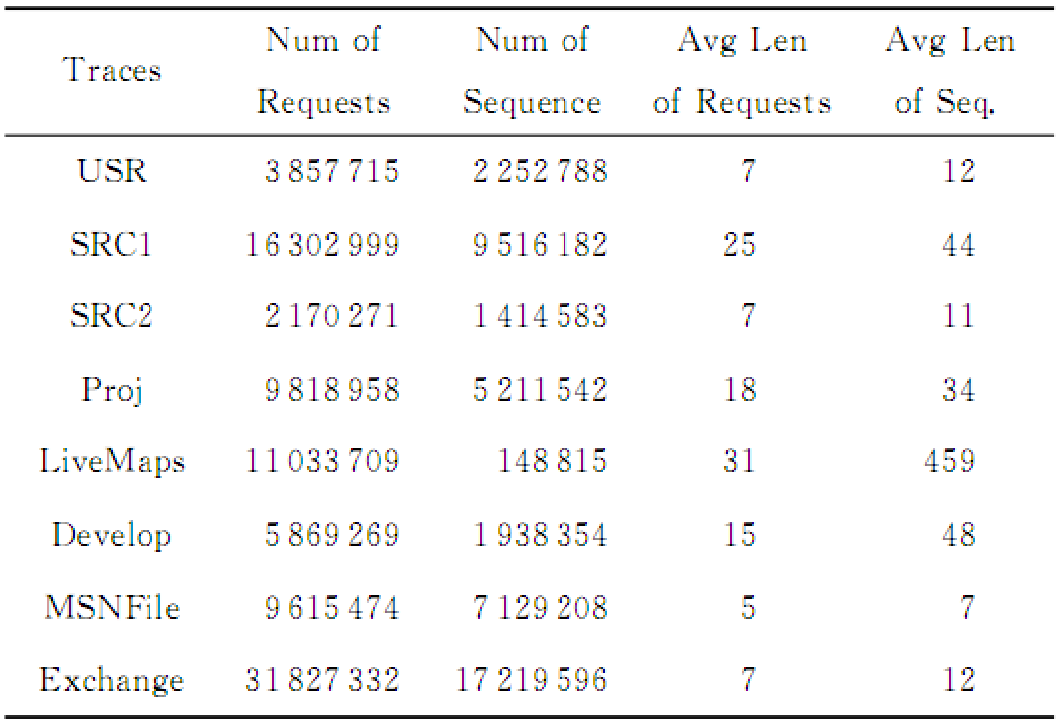
3. **Write data to the corresponding data segment.**

   In Step 2, the data of continuous free segments is simultaneously read to memory, these data will eventually be written to data segments accordingly. It seems like these data may be scattered across data segments of the hard disk, in fact, these data can be written to disk in order.

   When the file system assign space for an application, it is quite possible to assign a contiguous disk space, the application also visits data in order normally, in short, the same thread is often visiting the hard disk in order. However, due to the upper layer host may simultaneously runs multiple threads, which would visit alternatively, random I/O requests are generated as a result. In order to get the sequential requests, we need to filter these random request.

   For a particular thread, it may issue multiple writing requests in the same time, the data is written to the same free segment, or is written to the adjacent free segments. In Step 2 data in several consecutive free segments is written into memory, in memory, you can find multiple writing requests belonging to one specific thread, and these requests will be converted into a long writing request sequence. Figure 2(d) illustrates this process, three written requests Request 1, Request 2 and Request 3, in which Request l and Request 3 belong to the same continuous sequence, but is interrupted by Request 2. Three requests have been written to the free segment F3, when the data in F3 is read into memory, the sequence of Request 1 and Request 3 is recovered. In a word, in Step 3, when the data in memory is written back to disk, multiple writing requests are merged into one long sequence, this method reduces the total number of writing operations.

To support the discussion above, we have analyzed some standard working loads, the result is presented in Table 3, in which the first column indicates the eight working loads, and the second column indicates the number of requests, the third column indicates the number of sequences in each working load, the forth column calculates the average length of request, the fifth column calculates the average length of sequences. Since one sequence consists of multiple requests, the number of sequences is much less than the number of requests. The length of sequence is much larger than the length of requests. So the crux matter is to find out these continuous sequences to reduce the number of operations.



Traces	Num of Requests	Num of Sequence	Avg Len of Requests	Avg Len of Seq.
USR	3 857 715	2 252 788	7	12
SRC1	16 302 999	9 516 182	25	44
SRC2	2 170 271	1 414 583	7	11
Proj	9 818 958	5 211 542	18	34
LiveMaps	11 033 709	148 815	31	459
Develop	5 869 269	1 938 354	15	48
MSNFile	9 615 474	7 129 208	5	7
Exchange	31 827 332	17 219 596	7	12

Table 3 Sequential Requests Extensively Exist in Different Traces

In conclusion, we propose an I/O reorganization strategy which happens inside the hard disk, which has three steps, each step aims to avoid the head rescheduling, and to reduce the number of writing operations, thereby improving the performance of writing in order of hard disks.

In Step 1 above, if the data can be written to the appropriate data segment, since the address of the SSD corresponds to the address of data segment, no mapping table is needed to record this correspondence. On the contrary, if the data cannot be written to the corresponding data segment, it need to be written to the closest free segment to the magnetic head, in this case, a mapping table recording the data written to free segments is required, and it is stored in non-volatile memory.

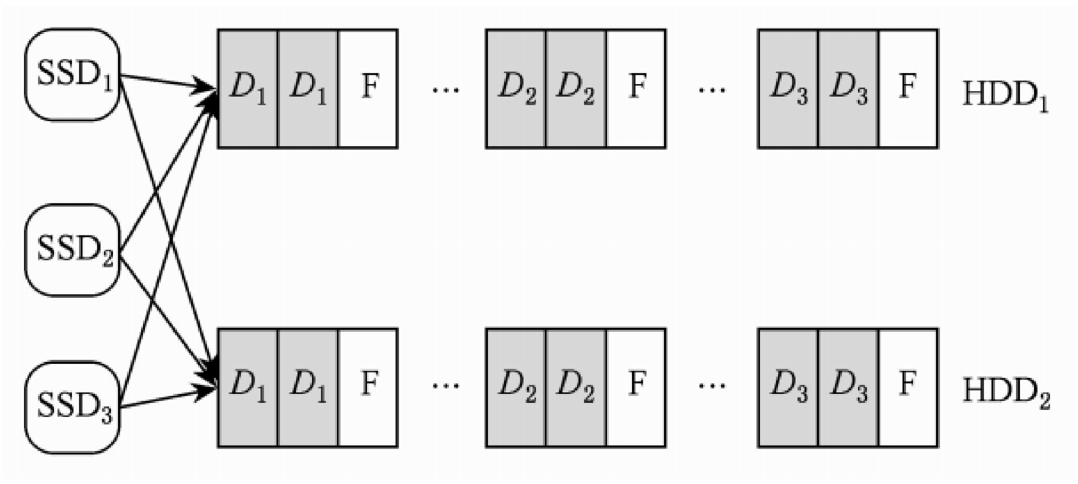
3 Bakcup Supported by Multiple Hard Disks

Section 2 walks through the writing optimization strategy inside the hard disks, which makes it feasible to back one SSD using one hard disk. But this 1 vs. 1 mechanism is not available for the following reasons:

1. Nowadays one single hard disk has much larger storage space than one SSD, for example, a certain hard disk usually is larger than 500GB, while the SSD is no more than 256GB, it causes waste of disk space.
2. There still exists vast gap in performance between the hard disk and SSD, if the upper layer host sends a sudden burst of writing request, the hard disk can hardly handle these data, leading to accumulation of large amounts of data in memory, if the memory is under heavy pressure, the SSD must be stopping to respond the user's writing requests. In order to solve this problem, we suggest collaboration between multiple hard disks for backup of multiple SSDs, which releases the pressure happens in single hard disk.

Collaborative multi-disk backup mechanism has two characteristics: first, the disk number less than the number of SSDs, this is because the larger disk capacity; Secondly, all the disks work together to provide backup for all SSDs, it means any disk space will be divided into a number of equal parts, which are provided to each SSD; any SSDs will divide their data into several equal parts, the fully related mechanism can distribute the sudden burst of writing requests on specific SSD dispersed into multiple disks, so that effectively suppress the peak working load on the single disk.

Figure 3 shows this multi-disk collaboration backup mechanism, as shown in Figure 3, two hard disks provide backup for 3 SSDs, each disk is divided into three parts, which support backup function for these 3 SSDs. Each part divides its addressing space into data segment and free segment according to the method described in Section 2.2, the data segment provides backup for specific SSD, as shown in Figure 3 the segments labeled D1 is for SSD1, the segments labeled D2 is for SSD2, free segments areshared by all SSDs. In this way, no matter what position the head is, any writing requests to specific SSD could be backed up to the free segment near the magnetic heads on time.



SSD1
SSD2
SSD3
D1 D1 F ··· D2 D2 F ··· D3 D3 F
HDD1
D1 D1 F ··· D2 D2 F ··· D3 D3 F
HDD2

Figure 3 Collaboration among Multiple Hard Disks

In fact, the multi-disk collaboration backup does not improve writing performance of the hard disks, what it does is only to distribute writing to one single SSD to multiple hard disk, so that multi-disk collaboration could easily handle peak writing to one single SSD. If all SSDs are under the burst writing pressure, collaborative multi-disk backup does not bring any benefits.

4 Experiment

4.1 Configuration

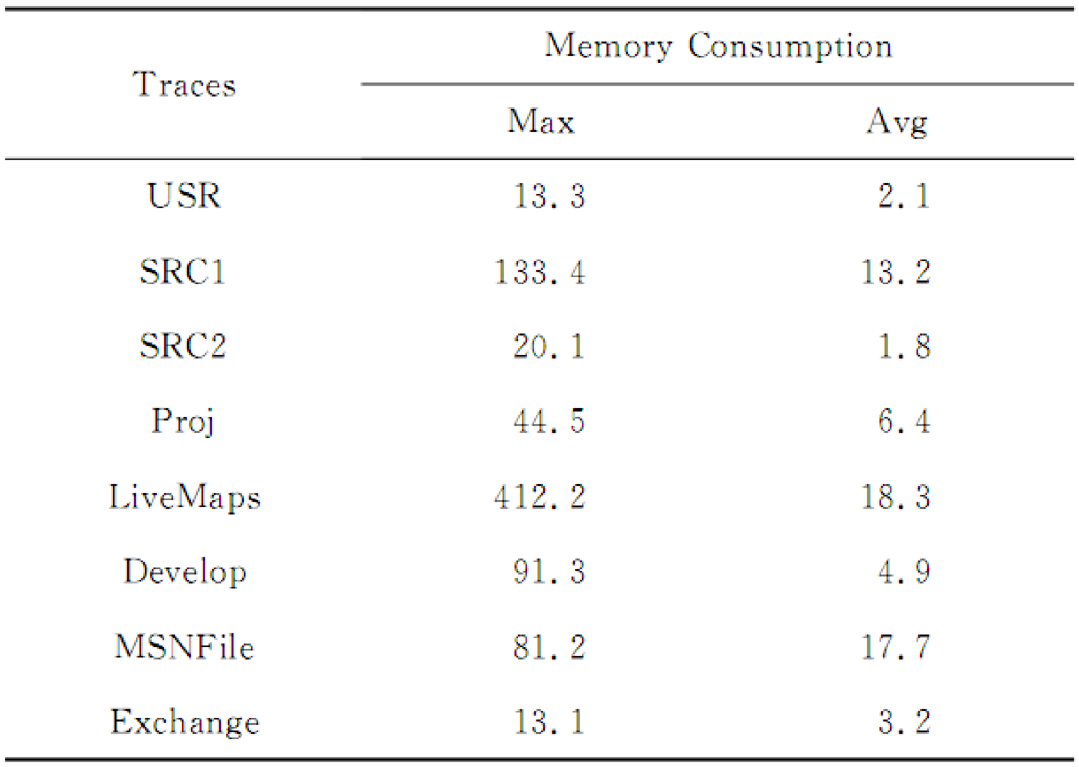
Our assessment of the achievement of the hybrid structure is implemented through a prototype system, using the equipment composed of the Intel 320 GB SSD and Seagate 500GB ST3500630A. First, we prove that the system is feasible. Next, the performance is compared with other systems. The input resources of system are some typical business working loads, as presented in Table 2, wherein SRC, USR and Proj are collected by Microsoft Research Institution of Cambridge; LiveMaps is collected through capturing I/O requests of LiveMaps backup servers during 24 hours. Develop is retrieved from the repository server. MSNFile is collected from the metadata of mail servers, Exchange collects the I/O requests during 24 hours. All these working loads are distributed by Narayanan [4][5].

The primary task of this experiment is to verify the feasibility of the system, which means the hard disks could support backup function for the SSD on time. This implies two aspects: the latency and bandwidth. The system solves the latency gap between the disk and SSD using a non-volatile memory, so the main point of this experiment focused on the hard disks could provide sufficient bandwidth. But we do not directly monitor the disk bandwidth, instead, we monitored the memory consumption of system. If the disk is able to provide enough bandwidth, the data could be quickly written to hard disks without accumulating in memory. On the contrary, if the disk bandwidth is not enough, a lot of writing data could congest in memory, therefore, the consumption of memory can accurately reflect whether there is sufficient bandwidth of hard disks.

4.2 Analysis about Feasibility of Single Hard Disk Backup

First in order to verify the feasibility of a single hard disk used for backup of a single SSD. We construct the backup system using one hard disk and one SSD, and send reading and writing requests to SSD while sending only writing requests to hard disk, meanwhile, we monitor the consumption of memory.

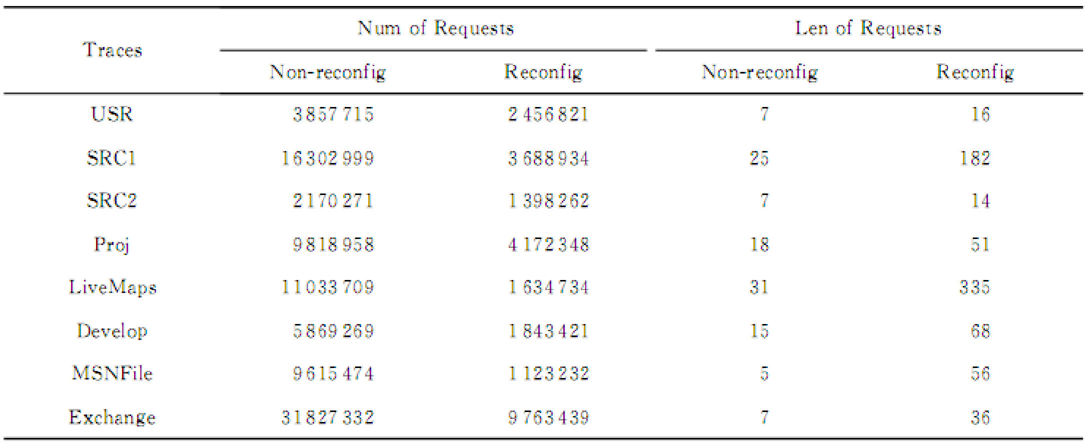
Working loads used for experiment are from the server, the strength of them may be very low, not enough to put pressure on the hard disks, also it could be intensively high to make single SSD hard to handle such intensive I/O requests. In both cases, we cannot verify that the hard disks can provide backup for SSDs. Therefore, we need to adjust the intensity of the loads. We use synchronization mechanism to send requests in order. The SSD could be working under a full load condition when a bunch of requests are returned immediately next bunch of requests are sent. If the hard disk could work well under this situation, our solution proved to be working.



Traces	Memory Consumption	
	Max	Avg
USR	13. 3	2. 1
SRC1	133. 4	13. 2
SRC2	20. 1	1. 8
Proj	44. 5	6. 4
LiveMaps	412. 2	18. 3
Develop	91. 3	4. 9
MSNFile	81. 2	17. 7
Exchange	13. 1	3. 2

Table 4 Memory Consumption of One Hard Disk vs. One SSD

The results are presented in Table 4, in which the last two columns indicate the max and average consumption of memory. As presented in Table 4, the average consumption of the memory is small, usually less than 20 MB. Generally speaking, a single disk can provide backup for a single SSD. However, for specific loads, the maximum memory consumption is tending to be large, e.g. LiveMaps, which consumes 400MB at most. These loads in certain time period caused a large number of burst writing operations, the writing data cannot promptly backed up to disk, and accumulate in the memory. Of course, the system can be equipped with large capacity memory to solve this problem, but large-capacity memory significantly increase the cost and power consumption. In Section 4.3 it will be proved multi-disk collaboration backup could solve it effectively. Table 5 explains why the disk can provide backup for the SSD. Because of the I/O reorganization inside the disk, the number of I/O has changed. Column 2 of Table 5 shows number before the reorganization and after the reorganization. This experiment showed that reorganization can significantly reduce the number of operation, so that reduces the working load of disks, therefore it is possible to provide backups for SSD on time. The 3rd column of Table 5 compares the average length of I/O before and after the reorganization. The results showed that, the increasing of average length after the reorganization leads to the decreasing of the number of I/O.



Traces	Num of Requests		Len of Requests	
	Non-reconfig	Reconfig	Non-reconfig	Reconfig
USR	3857715	2456821	7	16
SRC1	16302999	3688934	25	182
SRC2	2170271	1398262	7	14
Proj	9818958	4172348	18	51
LiveMaps	11033709	1634734	31	335
Develop	5869269	1843421	15	68
MSNFile	9615474	1123232	5	56
Exchange	31827332	9763439	7	36

Figure 5 Changes of Reconfiguration

The reconfiguration causes additional reading/writing requests. In Section 2, we will summarize the dispatching process inside the disk in three steps. In Step 1 original requests sent to disks are processed, in Step 2 and Step 3 reading/writing requests are generated inside the disk. Theoretically these extra reading and writing requests do increase the working load. However, as analyzed in Section 2, these additional requests are very long sequences. On the one hand, it does not significantly increase the number of I/O. On the other hand, sequential I/O requests are beneficial for the performance of the hard disks. Overall, I/O requests of the disks after the reorganization has decreased.

The feasibility of the system also depends on the share of the writing operation in a load, if the load is all writing operation, the load above disk and SSD are completely same, while the lower disk performance will obviously be the bottleneck of the system. Below the theoretical analysis proved that the system is also feasible under load of high percentage of writing.

Assume the system contains an SSD and a disk, the I/O capacity of disk is Pd, as for SSD, it’s Ps. The capability of I/O of disks determines the capacity of the entire system, if the load writing intensity is Pd, the disk works under full load, due to higher performance SSD could still respond to (Ps-Pd) reading requests. Therefore, when the load of a writing-reading ratio exceeds Pd/(Ps-Pd), the hard disk fails to back up.

Based on the analysis above, we believe that the proposed system is feasible based on two assumptions. First, the writing performance of SSD and disk differs a little, this is mainly because SSD has poor writing performance, with no obvious advantage compared with hard disk. Secondly, in the actual load, the writing operation normally has a small share. We analyzed eight kinds of loads, only one load has many writing operations though, the share is no more than 2/3.

Based on the analysis above, in real systems this mechanism is feasible.

4.3 Verification of Feasibility of Multi-disk Collaboration Backup

This section verifies the optimization of multi-disk collaboration backup compared with single disk backup. The experiment used three disks to provide backup for SSD, each SSD is divided into three equal parts, each part is backed up to three hard disks. The load in Table 2 is divided into two groups, USR, SRCI, SRC2 and Proj belong to Group 1, LiveMap BU, Develop, MSNFile and Exchange belong to Group 2, each group contains four different loads sending to 4 SSDs in system. Just like Section 4.2, we monitor the memory consumption. Results are displayed in Table 6.

Traces	Max Consumption			Avg Consumption		
	HDD1	HDD2	HDD3	HDD1	HDD2	HDD3
Group1	21. 6	27. 7	25. 3	19. 8	4. 4	5. 9
Group2	43. 3	31. 1	32. 2	6. 7	2. 3	4. 9

Table 6 Memory Consumption of Multi-disk Collaboration Backup

Table 6 shows the max and average memory consumption accordingly. Compared with one disk used for backup for one SSD, the memory consumption is small.

After reviewing Table 4 and Table 6, it’s obvious to know that the memory consumption does not decrease in multi-disk collaboration backup system. The reason is obvious, multi-disk does not reduce writing requests, it just distribute these requests to compress the peak load.

4.4 Comparison of Performance among Different Structures

Section 4.2 and Section 4.3 prove the feasibility of backup system consists of hard disks. This section will compare the performance between this system and other systems, including RAID1 completely based on SSD, RAID1 based on SSD and hard disk, RAID5 completely based on SSD, RAID5 based on SSD and hard disk.

In order to fairly compare the five strategies, each strategy comes with 4 SSDs storing user data. In detail that the RAID1 completely based on SSD contains 8 SSDs: 4 data disk and 4 mirroring disks, the RAID5 completely based on SSD contains 5 SSDs: 4

data disks and 1 checking disk (actually, the checking information is evenly scattered through each disk) The hybrid RAID1 contains four SSDs and four hard disks, HPDA contains four SSDs and two disks.

We choose four different memory size, and test the performance under the particular memory configuration. At the same time, for a fair comparison, other systems are equipped with 64MB memory.

Because each system contains four SSDs, in this experiment four different loads were sent to it. Like Section 4.3, we divide the load into 2 groups, of which each one contains four loads, respectively transmitted to four SSDs. The strength of load should be as large as possible in order to ensure SSD is always in full load state. Here we use IOPS (I/O per second) to measure performance. The experiment results are displayed in Table 7.

Traces	IOPS								
	SSD RAID1	Hybrid RAID1	SSD RAID5	HPDA	The Proposed Backup System				
					8 MB	16 MB	32 MB	64 MB	Three Disks
Group1	5 712	2 392	3 487	4 309	3 987	4 362	4 854	4 903	4 987
Group2	5 289	2 213	3 760	4 224	3 777	4 207	4 779	4 954	5 001

Table 7 The IOPS of Proposed Backup System

In the strategies mentioned above, RAID1 based on SSD gets the highest IOPS for its significant performance of reading. RAID1 based on SSD and hard disk gets the lowest IOPS, which is caused by the bottleneck of hard disk used for mirroring. The performance of hybrid RAID1 is similar with hard disk when responding to writing requests.

In contrast, the proposed hybrid backup system make writing requests sequential, and significantly reduces the number of I/O requests, thus obtains a better performance than HPDA. As shown in Table 7, when the memory capacity is 8MB, hybrid backup system achieved lower IOPS, which is due to the smaller memory capacity affects negatively when the disk handle many writing requests at the same time. When the memory capacity is 16 MB, the IOPS is almost the same as HPDA, when the capacity is 32 MB the IOPS improves further more. When the memory capacity is 64 MB, the IOPS does not improve any more. This is because 32 MB of memory is enough to deal with sudden writing requests.

5 Related Work

The traditional RAID5 writes data to each disk evenly. But for RAID5 solely based on SS, this would result in wear-out evenly on each SSD. When an SSD breaks up because of lifetime issues, other SSDs also may reach life limit, the possibility of error is high in this situation. So, traditional RAID5 mechanism cannot be applied directly to the SSD. Kadav et al proposed Diff-RAID[2]. They suggest sending checking information to each SSD not evenly to control the wear of each disk, so that the limits of the disks are reached at different time periods. Thus, when one disk reaches the life limit, simply replace it with a new SSD. This approach can avoid multiple disk failures, but for RAID based solely on SSD the cost is higher.

Griffin[6] utilizes hard disk as cache of SSD. This work is based on three aspects: 1) the performance of writing in sequence differs little between the disk and SSD; 2) the application involves a lot of the overwriting, 3) the application rarely read data written recently. If using hard disk as the cache of SSD when writing, the following effects could be obtained: 1) if the record takes the form of operation log, the disk cache can completely cache data written to SSD. 2) Writing requests could be reduced because of cache composed of hard disk, thereby extend the life of the SSD; 3) because data on disk is rarely read, this would not seriously affect reading performance. This work is based on an early SSD, the current SSD can do better.

I-CASH[7] is another hybrid storage architecture, it is mainly based on content locality while writing. When data is modified, normally only a small part is modified. I-CASH calculates the difference between the data before and after modification, then the difference will be compressed and stored to disk, thus reducing the SSD writing. While reading data. First, read the original data from the SSD, and then read the difference from disks to merge into the data. Due to the difference in value has been compressed and is small, it can be cached in memory to get good reading performance. Another paper [8] designed an SSD-based RAID controller. Due to the high performance of SSD, the traditional controller cannot be applied to RAID directly. The authors mention a multi-level controller structure, so that the controller is no longer a bottleneck.

6 Conclusion

This article proposed one kind of hybrid structure, which uses hard disk to back up SSD, with high performance and reliability. First, we propose one I/O reorganization strategy to reduce number of I/O while reading/writing, and make the I/O operations sequential. However, facing a large burst of writing, a single hard disk still hardly could meet the requirement. Therefore, we propose multi-disk backup for SSD. Our prototype system was analyzed and assessed thoroughly. The innovation embodied in this paper includes three aspects: 1) the hybrid structure of backup system for SSD, 2) use data reorganization to make I/O operations sequential, 3) use multi-block collaboration to solve the problem of sudden writing happened on one single SSD.

Reference

[1] Patterson, D. A., Gibson, G., & Katz, R. H. (1988). *A case for redundant arrays of inexpensive disks (RAID)* (Vol. 17, No. 3, pp. 109-116). ACM.

[2] Balakrishnan, M., Kadav, A., Prabhakaran, V., & Malkhi, D. (2010). Differential RAID: rethinking RAID for SSD reliability. *ACM Transactions on Storage (TOS)*,*6*(2), 4.

[3] Hu, X. Y., Eleftheriou, E., Haas, R., Iliadis, I., & Pletka, R. (2009, May). Write amplification analysis in flash-based solid state drives. In *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference* (p. 10). ACM.

[4] Narayanan, D., Donnelly, A., & Rowstron, A. (2008). Write off-loading: Practical power management for enterprise storage. *ACM Transactions on Storage (TOS)*, *4*(3), 10.

[5] Mao, B., Jiang, H., Wu, S., Tian, L., Feng, D., Chen, J., & Zeng, L. (2012). HPDA: A hybrid parity-based disk array for enhanced performance and reliability. *ACM Transactions on Storage (TOS)*, *8*(1), 4.

[6] Soundararajan, G., Prabhakaran, V., Balakrishnan, M., & Wobber, T. (2010, February). Extending SSD Lifetimes with Disk-Based Write Caches. In *FAST*(Vol. 10, pp. 101-114).

[7] Yang, Q., & Ren, J. (2011, February). I-CASH: Intelligently coupled array of SSD and HDD. In *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on* (pp. 278-289). IEEE.