12. Tang J., Sun J., Wang C., Yang Z. (2009) Social Influence Analysis in Large-scale Networks. *Proc. of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Paris, France, p.p 807-815.
13. Chen Z.Q. (2015) Analysis of Topic Evolution in Social Media Based on Co-word Network. *Information Science (In Chinese)*, 33(1), p.p. 120-125.
14. HU Y.L., Bai L., Zhang W.M. (2012) Modeling and Analyzing Topic Evolution. *Acta Automatica Sinica*, 38(10), p.p. 1690-1697.

NCache: A Cache Algorithm of SSD in Storage System

Heng Yang

School of Education Science, Nanyang Normal University, Nanyang 473061, China

Abstract
With the scale of the storage data increasing, the demand for data storage is higher and higher. The performance gap between the central processor and the disk is bigger and bigger so that People have to improve the performance of the disk. This is a big challenge to the storage system. Solid state disk (SSD in short) as a new storage medium, its performance and price are between the central processor and disk. The emergence of solid state disk is an opportunity to the storage system, which can narrow the performance gap between the central processor and disk. It is a hot research field in the current storage area. Based on the characteristics of SSD, we have proposed a new two levels cache algorithm called NCache which is used as a hot data buffer for the system. It can improve the hit rate of the cache and reduce the number of read and write times to improve the performance of the storage system. The buffer of random access memory is divided into the most recently used buffer and write buffer. We used Least Recently Used algorithm (LRU in short), which the latest data is based on to improve the system response time. The data written frequently is stored in the write buffer. The small write data can be aggregated into big write that can shorten the write numbers and extend the life of SSD. Test results show that the design of the two levels cache algorithm can work well. NLRU algorithm has advantages in improving the performance and reducing the number of solid state disk write. And with the use of solid state disk, the system can run faster than the system without SSD, greatly improving the performance of the system.
Key words: TWO LEVELS CACHE ALGORITHM, LRU, SSD, PERFORMANCE

1. Introduction

The explosive growth in social data is a severe challenge to on the traditional storage system. The digital universe Research Report released in the end of 2012 [1] shows that people will produce more than 40ZB data by 2020, which is equivalent to the amount of 5200GB per person on earth. At the same time, the report announces that all the data can be double every two years. The read and write requests will be more and more, which will form an impact to the storage system, so that the storage performance will be affected due to the response time of the request.

In recent years, high performance computing has gradually changed from the computation intensive to read and write intensive. The efficiency of reading and writing is very important to the system performance, and the requirement of I/O efficiency is increasing [2]. At the same time, the Internet service is also in high speed development, which requires a wide range of application of mass storage system to provide higher I/O efficiency. For the Internet application such as e-commerce and search engine, the user is sensitive to response time. It must handle a lot of operational requirements and ensure the user's response time be completed within the range of seconds [3]. Traditional storage systems must be improved to adapt to the environment of I/O.

SSD is invented as a new flash memory, which has many superior qualities compared with the traditional disk. First, there are not physical rotating parts, and the operation of read and write will no longer need to find ways, so that it can provide better read and write performance. It can not only have a lot of delay reach the millisecond in random read and write performance but also save energy [4]. Its price and performance are between dynamic random access memory (DRAM in short) and hard disk. Its performance is 100 times faster than the average disk, but the price of its unit capacity is ten times more than the average disk. The huge prices let many companies prohibitive. Although in recent years, its price has already been driven down, but it looks not realistic to replace the hard disk completely in the near future. The cost of using SSD is still more than the benefit brought by the performance increasing [5]. In any case, there is no doubt that SSD introduced into the enterprise storage architecture will improve the I/O performance of the storage system [6]. There are two ways to use SSD. The first one is to invest more money in exchange for good performance, the other one is to mix with SSD and disk to achieve the balance of performance and funds [7][8]. A lot of research in this area has achieved good results.

In this paper, we have set SSD as the second level cache in storage system to build a multilevel cache structure "DRAM-SSD", which can improve the system performance and reduce the system response delay. The design of the cache algorithm should be able to reduce the numbers of read and write, especially in reducing the numbers of write. The performance of SSD as read cache is very hard to improve. As a result, we can only use it as a write cache. While SSD has a written life, the impact of the overall system reliability, so in the design, the system should extend the life of SSD.

In general, LRU algorithm is used commonly to save data. The core idea of LRU algorithm is that the closer the data has been accessed, the more likely to be visited in the future. It will be organized into cache queue based on data entry time. The useless data is replaced from the beginning of the queue. It can be used to organize the data access time. The advantage of the algorithm is simple and low complexity, but when the work set is larger than the buffer's space, the cache will be no use. Let's image a scene that one buffer can only accommodate 100 data blocks, but the work set has 101 data blocks size. Use LRU management, whenever a block was eliminated from the access. In addition, when LRU algorithm scans large files, it perhaps cleans the buffer. This will lead to the original value of the data disappear.

The contributions of this paper are described as follows:

• SSD is added to a storage system, which constitutes a "DRAM-SSD" buffer layer. According to the characteristics of SSD and DRAM, a new caching algorithm is designed to achieve the goal of improving the I/O processing capability of the storage system. At the same time, we want the algorithm to achieve the goal of extending the life of SSD in the premise of performance improvement.

• We increased the write buffer in the RAM, which can reduce the write numbers to SSD and extend the write life of SSD.

• Performance test results show that the cache algorithm has advantages in improving the performance of the system. The performance can be improved more than 5 times compared without SSD. Write buffer test shows that the system can greatly reduce the impact of the SSD. It can be reduced by more than 40% of the write request, so that it can extend SSD's life well.

The rest of this paper is organized as follows. Section 2 describes related work in SSD. The architecture of our system and the related algorithms are introduced in Section 3. Section 4 is the experimental result and evaluation. Section 5 is the conclusion.

2. Related Work

SSD's chip is generally composed of one or more chips and each chip is divided into multiple planes. A plane is further divided into several thousand erase blocks, and an erase block is composed of 64 to 128 pages.

SSD appears as an alternative to the hard disk. In order to seamlessly with the existing system, SSD must provide a block interface. As the host, the SSD is seen as a hard device as hard disk [9]. The core of the SSD controller is Translation Layer Flash (FTL in short), which can simulate the SSD as an ordinary hard disk and provide logical address translation function. As shown in Figure 1, FTL receives the block level request from the host file system and processes them to produce a variety of control commands for SSD.

In order to solve the problems mentioned above, the FTL designers studied several techniques. The first one is address mapping technology, which maintains a dynamic change of address to the physical address of an address mapping table. Log write mechanism is the second technique. Each write operation let the pre-operation expire and add the new data to a clean page. The third skill is garbage collection, which will be active in fixed time intervals and focus the valid pages into a new erase block instead the old block. Because the write operation is often concentrated in a part of data page, which will lead to a number of blocks be worn earlier than the rest of the blocks. Loss balance mechanism can adjust the location of hot and cold data, so that the different positions of SSD have almost the same write numbers.

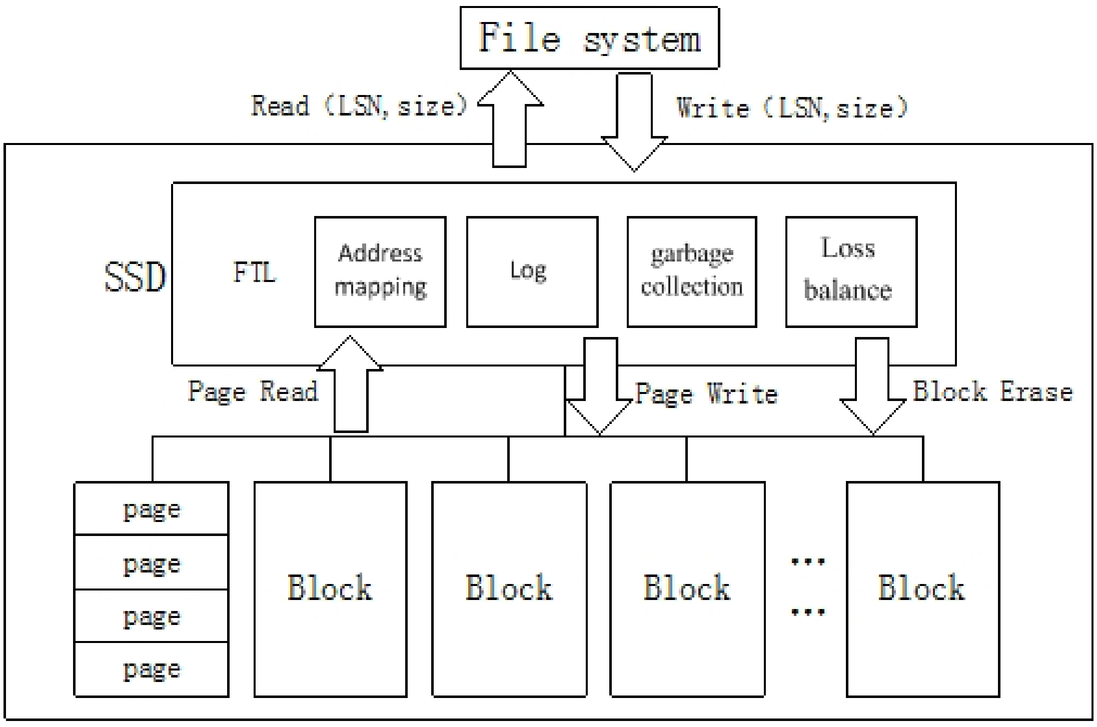

Figure 1. The figure of SSD structure

The lifetime of SSD is determined by three factors: the write request numbers, the size of the SSD space, the efficiency of garbage collection mechanism and loss balance mechanism. The less the write request is received by SSD, the longer the SSD’s lifetime is. SSD provides reserved space and the resource pool for the clean page, which can reduce the number of garbage collection, so that it can reduce the average write numbers to the pages and increase SSD's lifetime. The efficiency of the garbage collection mechanism and the loss balance mechanism, which are introduced in the extra write operation and erase operation. It is the write amplification effect, which is also a very important factor affecting SSD’s lifetime.

Most of the research has focused on the FTL's garbage collection mechanism and the loss equalization mechanism to improve the performance of the storage system. Taking into account the SSD finite write life, the frequent write operation not only reduces the performance of SSD, but also shortens the service life of SSD. It is not linear relationship between reducing write load and prolonging the life span. Reducing a half of the write operation can at least make the writing lifetime double.

3. The Load Balancing Algorithm

3.1. The Design of the System

In this chapter, we will introduce the architecture of our system.

The client is connected to the server through the network. After joining the SSD, the system structure is shown in Figure 2. Before the read and write requests are stored into the disks, they go through the buffer layer, which includes RAM and SSD. The function of the cache module is using RAM and SSD to cache partial data. After the data is accessed, it can be obtained directly from the RAM or SSD, without waiting for the data to be obtained from the disk. This will get a higher response rate.

Most of the requests are focused on a small portion of the data blocks, which is frequently accessed by the data blocks known as hot data. On the contrary, the data relatively low frequency access is called cold data. These hot data can be saved in the cache to reduce a lot of I/O request, which can greatly improve the system performance. Although the proportion of this part of hot data is only less than 10%, but the amount of hot data is also very large to RAM in big data system. It won't be a burden with the addition of SSD because the capacity of SSD can be much larger than RAM so that it can be satisfied with the space requirements of this part of the hot data

Data entered RAM at first, and they made up request in RAM. The current work set of data is generally cached in the RAM because part of read and write requests have time locality, that is to say, it will be accessed in the near future. The client also has a cache and the time locality of the request will hit in the client cache, so that the back-end server receives the request is not hit in the client's buffer. The time locality of the storage system is weak so that it is only needed to allocate some of the space in the RAM.

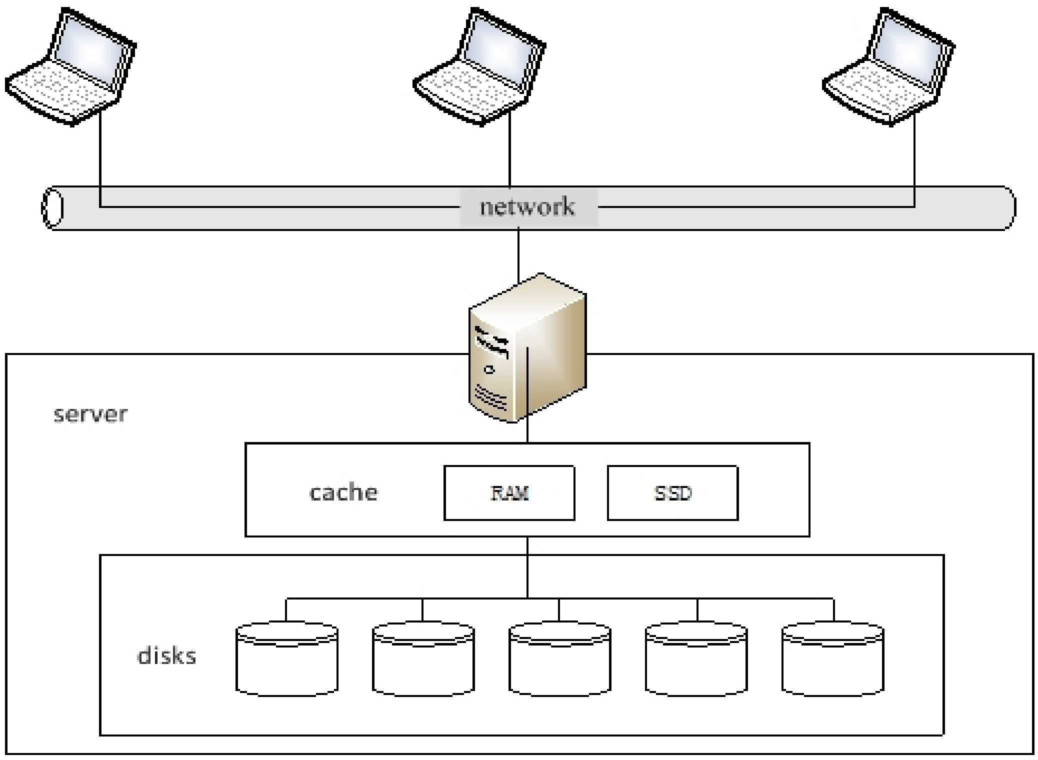

Figure 2. The algorithm description

The system can't import all the data from the RAM into the SSD, because a lot of data block will be accessed once. All the data imported into the SSD will cause the cache pollution, which will greatly reduce the use space of SSD. The writing and reading of these cold data also give SSD a lot of I/O burden. Based on the above reasons, we must design the corresponding algorithm to solve the problem.

A large number of hot data is cached in SSD. According to the statistics, a large number of read and write requests will focus on small part of hot data, so the SSD hit rate will be relatively higher. But this will bring a large number read and write operations for SSD. The hit is good for the system performance, but too much read and write operations will be adverse to SSD. Because the SSD can be written in a limited number of times, read and write frequently will let SSD's life be shorter.

The system also needs to solve the problem of writing life of SSD, which only can be solved by reducing the write numbers of SSD in the device layer to extend the life of SSD. In order to increase SSD's write life, a write buffer is allocated to the RAM, and the buffer is sent to the lower layer. At the same time, we notice that this write buffer can reduce the weakness of read and write performance inequality. It can improve the performance of the system that the system reduces the write request to these two devices.

3.2. Algorithm Description

According to the statistics, a large number of read and write requests will focus on a relatively small proportion of hot data. The data blocks written more times will be cached by the write buffer module, which always intercepts a large number of requests send by the upper layer. The data block stored in the write buffer is the data written frequently. Because of the limited space in the buffer, the most frequent data blocks can be cached. In the design of this algorithm, we designed a new algorithm called NCache to replace LFU cache algorithm, which is achieved directly by a number access array and a priority queue. The priority queue is implemented by the heap, on the top of which is the element of the smallest number of accesses.

Write buffer algorithm creates an array to record the write numbers of all data blocks in the RAM. The block is large in the cache module and the two blocks of access frequency is not necessarily the same even with the same number of times, so that we can not consider the only factor of the write numbers. For example, there are two blocks with the same write times and the write times are 12. We divide the blocks into four units. The grey block means that the block has already been written. In this figure, we can see that two units have been written in block A and one unit has been written in block B. We know that the write numbers of a block is the sum of the write numbers of units. The write numbers of block A should correspond to that of block B, but the write numbers should share equally to all the unit that have ever been written. By this way, the write frequency of block A is six times while that of block B is 12. We can make conclusion that the write in block B should be more frequent than write in block A.

In the design of write buffer queue, we divide the write numbers to one unit into the units that have ever been written. We set the calculated results as write frequency. Write buffer queue is achieved with a small top stack, in which the comparison function is function to compare the block write frequency at first.

When data block will be eliminated in the write buffer, the block with larger size will be eliminated at first if the two blocks is the same write frequency. This is because the write buffer can merge small write into capital big write. With time passed, small data block is also getting bigger after multiple read and write operations in write buffer. Eliminate the larger blocks can not only get more free space but also reduce the write numbers to SSD effectively. The time cost to write a 128K data block is the same as that to a 16K block.

It can extended the write life of SSD to reduce write request by write buffer. In addition, SSD has the disadvantage of write amplification, that is due to the introduction of additional data copy and erase operation by the SSD firmware in the FTL to achieve the load balance, garbage collection and other functions. It can not only reduce the write operation on SSD, but also slow down the additional write amplification to get good performance that effect write buffer merge small write request into big the write request.

It can reduce the negative effect of the read and write performance asymmetry by reducing the write operation. Data block will be turned from small trunk into large trunk in write buffer so as to reduce the additional reading overhead and improve the system performance.

3.3. The module of the system

The whole system is divided into five functional modules, including global management module, target module, cache management module, control module and initiator module, which are shown as Figure 3. The following is a brief introduction of these five modules.

Global management module can manage logical disks and physical disks. It can interact with the user state of the web page to complete the creation of the disks and assign functions for the client. The module also manages the information of the size, the numbers and the capacity of the logical disk, which can be accessed by other modules through the interface of the global management module. The global management module manages the mapping structure of physical disks and logic disks to provide the interface for the target module to map the logical disks. The size of the trunk is generally set to the default value 64KB.

Target module is the server side of the SCSI protocol, which receives read and write requests from remote clients and other control commands. The client's read and write requests are organized into Bio, which contains a number of physical addresses of discontinuous pages. In this module, one request is divided into several small read and write requests, each of which contains a page in the Bio. After division, the requests will be send to the next layer of cache management module.

The main function of cache management module is to manage RAM and SSD cache space. They cache the hotspot data, which can be directly read and written in the buffer layer to improve the I/O performance of the whole system. When the read request is not hit or need to synchronize the data blocks, the request is sent to the control module. In this layer, SSD is used as a bare disk, so that read and write SSD is to interact with the initiator module directly.

Read and write functions are both realized in control module, which provides read and write interface to cache management module. According to the different level, it send read and write requests to initiator module to complete the operation of the actual physical disk.

Initiator module interacts with the actual physical disk. At the beginning of the system, this module scans all the physical disks on the server to get the disk capacity, channel number and other information, which can provide the interface to global management module to add disks for the system. In the system, SSD is also used as a common physical disk.

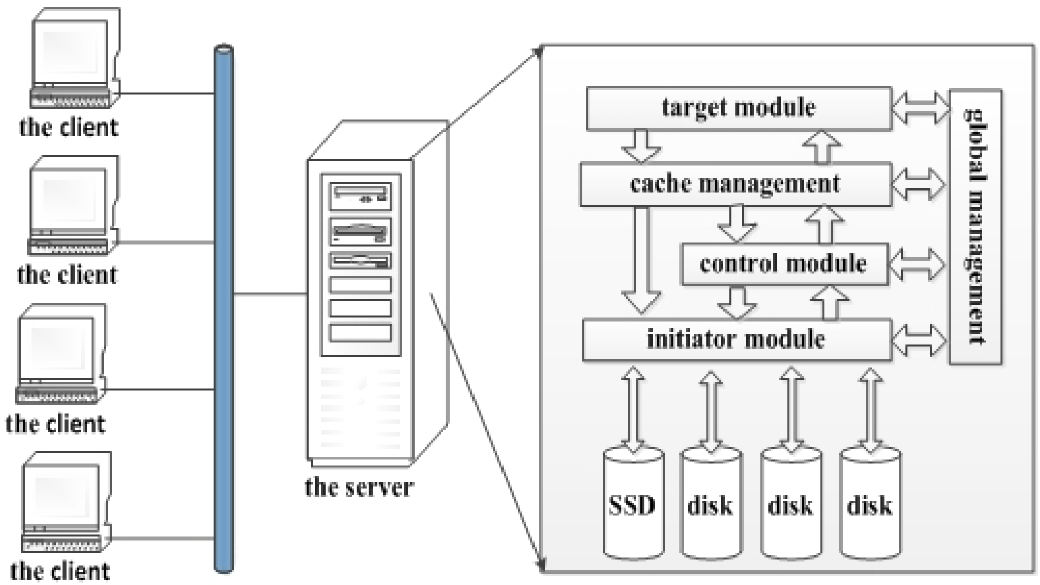

Figure 3. The module of the system

4. Experimental results

4.1. Experimental Setup

The hardware configuration of the server is shown in table 1. The server has a system disk, three data disks and a SSD.

Table 1. The hardware configuration of the server

name	value
CPU	Intel(R) Xeon(R) CPU E5606 @ 2.13GHz × 8
memory	Hyundai 4GB × 4
system disk	Seagate 320GB 7200rpm
data disk	Seagate 1TB 7200rpm
SSD	Seagate 600 SSD 6Gb/s 120GB
operation system	Fedora release 14
kernel version	2.6.35.6-45

4.2. Test Result

At First, we test the performance of the storage system in two cases: with the use of SSD cache and without the use of SSD. We set time one hour. We test in two groups, one sets RAM size is 512MB, the other fixed RAM size 1GB. We can get the performance curve shown as Figure 4. In the figure, the SSD capacity is 0, which means that the system does not contain SSD. Figure 5 is acceleration ratio curve. The acceleration ratio is defined as the ratio of the performance of the system using the SSD cache and without the SSD under the same test conditions.

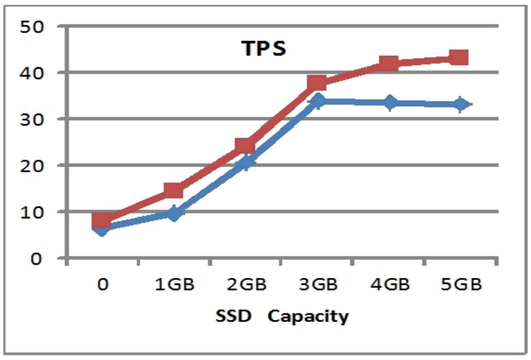

Figure 4. The performance curve

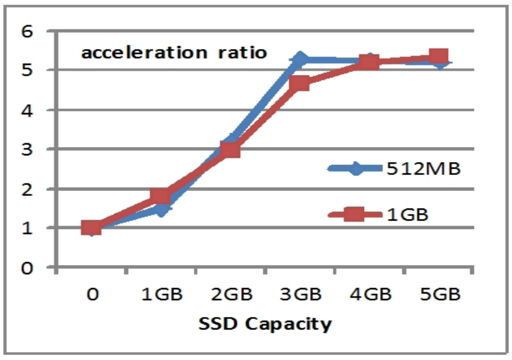

Figure 5. Acceleration ratio curve

From Figure 4, we can conclude that TPS increases with the increase of SSD capacity. At the first time, it grows significantly, but when the SSD capacity increased to a certain extent, it increases very slow. It will tends to be smooth gradually and reach the peak value of the performance.

The acceleration ratio curve can be seen in Figure 5. With the increase of SSD capacity, the performance acceleration effect is very obvious. When RAM is 512MB or 1GB, the maximum acceleration ratio is very high.

5. Conclusions

Read and write performance is asymmetric in the storage system. It is very limited to use RAM as a buffer to improve the performance. Adding SSD on the storage system as a buffer can improve the performance of the storage system. It can extend the SSD life when we add a write buffer in RAM to reduce the write numbers of SSD. This method can achieve the goal of improving the performance of the system and increase the reliability of the system. The experimental results show that the algorithm has reliable performance.

There are lots of works to do in the future which are described in the following directions. To implement the caching scheme, there are some key parameters directly hard encoding, or need to be set before running. These parameters are set in advance according to different operating conditions, so that the adjustment of these parameters will take the user higher requirements. It will be better if these parameters can be realized in different operating environment. There is also a vast space in refinement of the task classification and full support of various tasks classifications.

References

1. Xavier Jimenez, David Novo and Paolo Ienne (2014) Wear Unleveling: Improving NAND Flash Lifetime by Balancing Page Endurance. *Proc. Conf. on the 12th USENIX Conference on File and Storage Technologies*. California, p.p. 47-60.
2. Chen F., Koufaty D A., Zhang X. (2011) Hystor: making the best use of solid state drives in high performance storage systems. *Proc. Conf. on the international conference on Supercomputing*, p.p. 22-32.
3. Lee S W., Moon B., Park C., et al. (2008) A case for flash memory ssd in enterprise database applications. *Proc. Conf. on the 2008 ACM SIGMOD international conference on Management of data*, p.p. 1075-1086.
4. Payer H., Sanvido M A., Bandic Z Z., et al. (2009) Combo drive: Optimizing cost and performance in a heterogeneous storage device. *Proc. Conf. on the First Workshop on Integrating Solid-state Memory into the Storage Hierarchy*, p.p. 1-8.
5. Andersen D G., Swanson S. (2010) Rethinking flash in the data center. IEEE micro, 30(4), p.p.52-54.
6. Oh Y., Choi J., Lee D., et al.(2012) Caching less for better performance: Balancing cache size and update cost of flash memory cache in hybrid storage systems. *Proc. Conf. on the 10th USENIX Conference on File and Storage Technologies*, p.p. 23-28.
7. Wu G., He X. (2012) Delta-ftl: improving ssd lifetime via exploiting content locality. *Proc. Conf. on the 7th ACM european conference on Computer Systems*, p.p. 253-266.
8. Chen F., Luo T., Zhang X. (2011) CAFTL: A Content-Aware Flash Translation Layer Enhancing the Lifespan of Flash Memory based Solid State Drives. *Proc. Conf. on the 9th USENIX Conference on File and Storage Technologies*, p.p. 77-90.
9. Pritchett T,. Thottethodi M. (2010) SieveStore: a highly-selective, ensemble-level disk cache for cost-performance. *Proc. Conf. on ACM SIGARCH Computer Architecture News*, p.p. 163-174.