Survey on different cache replacement algorithms

[1]Shruti Kudagi, [2]Dr.Naveenkumar Jayakumar

Computer Engineering
BV(DU)COE
Pune, India

***Abstract*: The SSD (Solid State Drives)shares and common interface between logical and physical path to HDD(hard disk drives).New SSD are equipped with Flash memory and they are on-board chip .They are having multiple Flash chips on single board which acts like cache . The flash chips are non-volatile .Basically there are two types of flash chips which uses NOR or NAND gates. But NAND based flash in SSD are good performance as compared to NOR based flash because they more expensive and hard to erase, read and write new data. So to manage the access of data from cache and data in flash memory different cache replacement algorithm are used so that the data retrieval is faster and access time required is less. Different algorithms are LFU (Least frequently used), LRU (Least recently used), Adaptive replacement cache (ARC).Most Recently Used (MRU), these all algorithm show different result in different scenario. It all impacts on performance of system. Other issues like software issues ,Hardware issues which includes garbage collection, file access rights, parallelism, workload balance, Blocks, flash chips, I/O request, cost where performance is depend on. In this paper we are focusing only on cache and flash chips, its access related to performance. The LCR algorithm focuses more on workload balance and access time both where other algorithm do not consider. To overcome problems of read, write, erase, I/O request some consider load balance of flash chip to resolve overhead and access problem we use different cache replacement algorithm. The cache penalty is consider by LCR algorithm which is newly introduce in algorithm and not used by other traditional algorithms. In this paper there is survey on different cache replacement algorithm so that we can get to know the better algorithm suitable for application so that performance can be increase. The survey can help us to get or make more advancement in algorithm which will consider some more parameters and will enhance performance reducing other overheads.**

***Keywords*: ARC, cache replacement, LCR, LFU, LRU, NAND, NOR, Solid State drives**.

I. Introduction

Flash memory is bringing more changes in the of storage system from many years. It is more widely used now- a-days in many datacenters, computers and enterprise storage systems. The Solid state drives actually perform as a alternative to hard disk drives (HDD) .The hard drive uses magnetic coating by countering to that the solid state drives perform better in random inputs and outputs. Solid state drive mainly uses less power than HDD but cost wise SSD are more costly which actually cannot afford in place of HDD. So they are mainly used in cache. This can manage the cost. SSD shows its good working on improving performance, but it could suffer from resolution issue when arranged as disk cache. Sustainability capacity of Flash memory is only limited and can handle less erase/write cycles. Erasing continuously this effects the SSD lifespan. Such defects can be taken cared by different software.

Different cache algorithms are basically focused on buffer cache which reside in RAM. Many of them, cache replacement algorithms depend on hit ratio to maximize their performance by utilizing cache .There are large number of algorithms that have been introduced. The papers focuses on survey of different cache replacement algorithms which perform differently on basis of problem. In this paper, we do survey on different algorithm on SSD based disk cache, algorithms like named Lazy Adaptive Replacement Cache(LARC), (Least recently used)LRU, Least frequently used (LFU).Adaptive replacement cache, Most recently used(MRU). Load aware cache replacement algorithm. In this paper survey is done on these algorithms only.

II. Background

A. Flash Memory and SSD

Flash memory comes under electronic device which is use to stores data in cells. This is called as floating-gate transistor. Basically these cells are used to program to represent different states to show one or more bits. Mainly there are two types of flash memory. They area NOR and NAND. In NOR gate basically the use byte to word level addressing and enhance faster read. Nand is cheaper and density to store data is high

Mainly SSDs use NAND flash memory for the storage medium. These NAND flash based memory can be categorized into Single Level Cell (SLC) and Multi Level Cell (MLC) flash. In SLC flash memory cell it stores only one bit. In MLC flash memory cell can store up two bits or even more than two bits. In NAND Flash memory cell is organized into blocks. 64 to 256 pages are included in each cell. Each page is capable of storing 2KB or 4KB data and a metadata. Each read and write operations are done in unit of page and when erase take place it goes in to block unit. Each block must be handled properly so that they should not been over

written. Every block can erase finite number of times. The MLC flash memory has capacity to handle 10,000 erase per cycles. In SLC they can upto 100000 erase cycles.
As challenging with hard disk, the SSD must maintain its perfor-mance advantages even with low cost arrangements, such as using MLC flash. Unfortunately, both performance and durability of flash memory decrease as the storage density increases
As challenging with hard disk, the SSD must maintain its perfor-mance advantages even with low cost arrangements, such as using MLC flash. Unfortunately, both performance and durability of flash memory decrease as the storage density increases [9].In MLC the write speed is three times lesser than SLC flash based memory and this makes the writes more expensive in MLCs .Erase cycle in MLC flash is ten times smaller than that of SLC flash. This concludes that the impact of write on MLC is more important than on SLC. To avoid directly writes to requested user, the garbage collection can create extra write and the writes can be reduced through designing proper architecture.

B. SSD Cache Models

There are mainly two models are used for SSD based disk cache. In this model one shows that SSD used is an extension to memory. Basically in this model RAM and SSD are to be single cache tier in HDD. In the other model, SSD is used as extended disk. The SSD nearly lie below the standard block interface SSD lies beneath the standard block interface and this depicts the second cache level. It is servers to be transparent to components lying in the system.

Due to the comprehensive property of multi-level cache the long disk model is often lower to the other one on collective hit rate. Implementing the prolonged memory model usually involves changes of the application itself or operating system. This is very costly. Therefore, the mainly used model is extended disk model is more in production systems. And mainly researcher's focus on this focused on it in this research.

C. Problems with Cache Algorithms

Exhaustive study on cache management has large number of algorithms. To increase the usage of cache devices, a best algorithm wants always to keep the popular blocks in cache. Although existing algorithms are mainly depend on two experiential norms. The first is called as temporal locality, that is recently used blocks are main likely to be used again for future. The other one is tilted regard of blocks that is some blocks are more frequently accessed than others. Accordingly, these two classical algorithms are suggested, called as LRU and LFU. LRU is mainly used in simple work of production systems and gives O(1) as the overhead. It has drawback of less performance for many workload with weak locality. The example basis on this are: which scans on large data set

This can easily eradicate cache space and occupy it with one time retrieve the blocks. LRU is extremely susceptible to this. In a share storage system, when one of the users starts for scanning the frequently blocks are been accessed by others. It will pushed oout and try to increase response time intensely. Second example includes the loop in file are slightly larger in size of cache in this the Least recently used will try to reaccess .But the LRU always try to evict it and this degrade the performance.

LRU is inadequate to manage with these access patterns .Since it simply disregards the regard of blocks. Frequently accessed blocks can be incorrectly be replaced by infrequently used ones. Several refined algorithms are been proposed to solve such problem. They are as FBR [2], EELRU [3], 2Q [4], LIRS [5], MQ [6] and ARC [7]. These algorithms can easily identify infrequently accessed blocks and evict them earlier. Thus they increases hit rate by positioning popular blocks in cache for a longer period of time. Taking ARC an example, It divides cache blocks in two groups which according their access frequency. For One-time accessing blocks are stored in T_1 and other blocks in T_2. A threshold P is used to give the limit the length of T_1. When the length of T_1 is larger than P , blocks are always evicted from T_1. Otherwise, blocks are evicted from T_2. The value of P is vigorously adjusted. As a result, blocks in T_2 roughly have a higher priority thus stay longer in cache.
However, none of the algorithms takes the for write endurance in SSD. When functional to SSD based disk cache, the susceptibility of LRU not only decreases the performance, but also suffers pointless write traffics to SSD, limitation its lifetime. As for ARC, evicting one-time accessed blocks from T_1 improves hit rate, but the write resolution issue remains the same . These blocks will not been accessed during their residence in cache, they should not be written to SSD directly at all. If these blocks are identified and keep out of cache, we can enhance the hit rate and reduce SSD write traffics at the same time. This encouraged us to push the research forward and design a new SSD friendly cache algorithm.

III. Literature survey

The following study shows the comparison between the different cache replacement algorithms. Each paper discusses certain development and issues regarding different algorithm. In this paper only selected algorithms are studied which will help us ton anlyze better algorithm among all.

[1]Paper on focuses on LRU policies which help us to evict the blocks looking to the past data. Cache updating is mainly concern so that cache hit is more. The MRU has also has same issue as compared to LRU but it has little better results. So MRU and tree based cache heap onject is better performance than LRU. [2]In this paper, author has sproposed algorithm RARC where advancement to ARC. Which out performns than FIFO and LRU algorithm. In our paper we have taken glance of ARC to understand the algorithm.[3] This paper helps us to understand the core working of SSD, there models and performance .They have also discussed many issues related to algorithms which have random write problems and work load distribution.[4]In this paper they have concluded that this algorithm has improved many algorithmic issues of existing cache replacement algorithm. In this paper

they have given more priority to cache which are overhead and they should be clean out as per algorithm design. This also show the good performance than LRU cache replacement algorithm.[5] In this paper, ARC shows the better performance than LRU by capturing the recency and frequency both are considered . The ARC outperforms more than LRU. Table has following parameters on basis of that we conclude the best algorithm among five algorithms which outperform different from each other's.
'

Table. 1. Survey on algorithms

Algorithmic parameters	LRU	LFU	Adaptive replacement cache	MRU	LCR
Cache hit	Yes	Yes	Yes	yes	Yes
Cache miss	more	average	less	average	Less
Cache penalty	No	No	Less	less	Yes
Workload consideration	No	Less	More	no	Yes
Cost	High	Not high	High	Very high	Reduced
Read and write flow	Efficient	Reads are efficient	Read and write both are average	Read are more	Efficient
Response time	Improved	Average	Improved	improved	Improved
Effect of Cache size	Improved	average	Can effect	Can effect	Improved

IV. Conclusion

The above survey done considering different cache replacement algorithms with their parameter depicts that the LRU performs less than all the algorithms. The ARC algorithm combines the frequency and recency which gives average good result than the LRU. The which outperforms among all the LCR which has developed garbage collection and enhance the algorithm to increase more read and write hits which gives better result. Though it has complex architecture it has reduced designing cost compare to other algorithms

References

[1] Burhan Ul Islam Khan et.al, A cpmputation efficient P-LRU based optimization cahe heap object replacement policy , IJACSA, Vol. 8, No 1, 2017

[2] Feng Ye Jianxi Chen et.al , A regional popularity aware cahe replacemnet algorithm to improve the performance and lifetime of SSD cache disk cache, 2015, IEEE.

[3] Feng chen et.al , Understanding intrinsic characteristics and system implications of flash memory based solid state drives , 2009

[4] Caiyin Liu et.al,LCR : Load aware cache replacemnet algorithm for flash based SSDs, 2018, IEEE

[5] Qiabin Xia et.al , High perforamce and edurable cache management for flash based read cache, 2015, IEEE

[6] Desai, P. and Jayakumar, N., AN EXTENSIBLE FRAMEWORK USING MOBILITYRPC FOR POSSIBLE DEPLOYMENT OF ACTIVE STORAGE ON TRADITIONAL STORAGE ARCHITECTURE.

[7] Desai, P.R. and Jayakumar, N.K., A Survey on Mobile Agents.

[8] Jayakumar, N. and Kulkarni, A., 2017. A Simple Measuring Model for Evaluating the Performance of Small Block Size Accesses in Lustre File System. Engineering, Technology & Applied Science Research, 7(6), pp.2313-2318.

[9] Rishikesh Salunkhe, N.J., 2016. Query Bound Application Offloading: Approach Towards Increase Performance of Big Data Computing. Journal of Emerging Technologies and Innovative Research, 3(6), pp.188-191.

[10] Salunkhe, R., Kadam, A.D., Jayakumar, N. and Joshi, S., 2016, March. Luster a scalable architecture file system: A research implementation on active storage array framework with Luster file system. In 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) (pp. 1073-1081). IEEE.

[11] Jayakumar, N., Iyer, M.S., Joshi, S.D. and Patil, S.H., A Mathematical Model in Support of Efficient offloading for Active Storage Architectures.

[12] Naveenkumar, J. and Joshi, S.D., 2015. Evaluation of Active Storage System Realized through MobilityRPC.

[13] Jayakumar, N., Bhardwaj, T., Pant, K., Joshi, S.D. and Patil, S.H., A Holistic Approach for Performance Analysis of Embedded Storage Array.

[14] Naveenkumar, J., Makwana, R., Joshi, S.D. and Thakore, D.M., 2015. Performance Impact Analysis of Application Implemented on Active Storage Framework. International Journal, 5(2).

[15] Naveenkumar, J. and Joshi, S.D., 2015. Evaluation of Active Storage System Realized Through Hadoop.

[16] Zaeimfar, S.N.J.F., 2014. Workload Characteristics Impacts on file System Benchmarking. Int. J. Adv, pp.39-44.

[17] Jayakumar, D.T. and Naveenkumar, R., 2012. SDjoshi,“. International Journal of Advanced Research in Computer Science and Software Engineering,” Int. J, 2(9), pp.62-70.

[18] Suryawanshi, A.U. and Naveenkumar, J., PRIVACY-PRESERVING FOR A SECURE DATA STORAGE ON CLOUD USING PUBLIC AUDITING TECHNIQUE.

[19] Suryawanshi, A.U. and Kumar, J.N., A Multiuser Model of Privacy-Preserving Auditing for Storing Data Security in Cloud Computing.

[20] Naveenkumar J and Raval K.S, Clouds Explained Using Use-Case Scenarios.

[21] Sagar S Lad, S.D Joshi, N.J, Comparison study on Hadoop’s HDFS with Lustre File System. International Journal of Scientific Engineering and Applied Science, 1(8), pp.491–494, 2015.

[22] Jayakumar, N., Bhardwaj, T., Pant, K., Joshi, S.D. and Patil, S.H., 2015. A Holistic Approach for Performance Analysis of Embedded Storage Array. Int. J. Sci. Technol. Eng, 1(12), pp.247-250.

[23] Kumar, N., Kumar, J., Salunkhe, R.B. and Kadam, A.D., 2016, March. A Scalable Record Retrieval Methodology Using Relational Keyword Search System. In Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies (p. 32). ACM.