ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

Volume 3, Special Issue 3, March 2014

2014 International Conference on Innovations in Engineering and Technology (ICIET'14)
On 21st & 22nd March Organized by

K.L.N. College of Engineering , Madurai, Tamil Nadu, India

Efficient Cache Utilization of Disk Array Failures

K.Shamili[1],P.Golda Jeyasheeli[2]

[1] Department of Computer Science and Engineering, Mepco Schlenk Engineering College , Sivakasi, India.

[2] Department of Computer Science and Engineering, Mepco Schlenk Engineering College , Sivakasi, India.

Abstract— Cache memory plays an important role in computer Architecture. The cache memory is the smallest and fastest memory, which reduces the average access to the storage device i.e. the RAID device. The existing cache replacement algorithms work effectively in fault free mode, but with poor performance. When one disk in the disk array fails, the request to the faulty disk is directed to the other non-faulty disks in the RAID disk to reconstruct the lost data. This increases the number of requests to the non-faulty disk. This decreases the performance and reliability of the system.

To improve the performance, Victim Disk First (VDF) cache replacement algorithm is used. This algorithm retains the blocks of faulty disk in the cache.ie the blocks of faulty disk are not selected for replacement. This reduces requests to the other non-faulty disk. The VDF-LRU and VDF-LFU cache replacement algorithms are employed to replace the blocks of the non-faulty disks when the cache is full. VDF-LRU efficiently improves the performance by not selecting the blocks that are faulty.

Keywords— VDF, VDF-LRU, RAID, CACHE MEMORY, DISK ARRAY.

I. Introduction

TO reduce the number of requests to the low-level storage device, such as disk arrays, a cache[2][3][11] is widely used and many cache algorithms exist to reduce the long disk latencies. These cache algorithms work well for disk arrays under normal fault-free mode. However, when one disk in a disk array fails, the redundant arrays of independent disk (RAID)[9] may still work under the faulty scenario. The cost of a miss to faulty disks might be high compared to the cost of a miss to other non-faulty disks. Existing cache replacement algorithms cannot capture this difference because they treat the underlying (faulty or non-faulty) disks in an array the same.

II. System Design

We take an example as shown in Fig. 1, which illustrates two different cache miss situations in a storage subsystem composed of RAID level 5 which is parity-based with one faulty disk. As shown in Fig. 1a, the missed data resides in the faulty disk. The RAID controller accesses the non-faulty disks to fetch all data and parity in the same stripe to regenerate the lost data. Therefore, to service one cache miss, several read requests are needed depending on the RAID organization. However, if the missed data is in a non-faulty disk as shown in Fig. 1b, only one read request to the corresponding non-faulty disk is generated.

Therefore, in parity-based disk arrays under faulty conditions, a miss to faulty disks is more expensive than a miss to other non-faulty disks. Based on this observation, a new cache replacement strategy, named Victim (or faulty) Disk(s) First cache (VDF), to improve the performance and reconstruction duration. The basic idea is to treat the faulty disk more favorably.

VDF retains the blocks of the faulty disk in cache. When one disk in the disk array fails, the data of the failed disk is retained in the cache memory. When the cache is full, cache replacement algorithm VDF-LRU is applied to the blocks of the faulty disk and the faulty disk blocks are not chosen for replacement. By retaining the blocks of the faulty disk in cache, the number of requests to the other non-faulty disk is reduced which in turn improve the performance and speed up the reconstruction of the lost data.

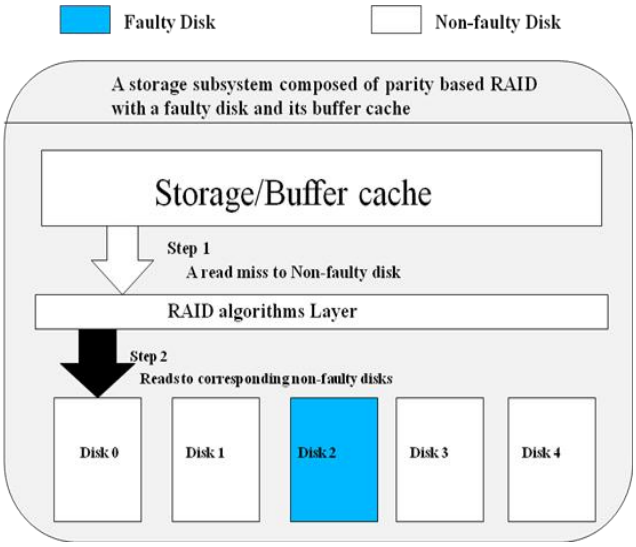

Fig 1a: RAID 5 working in fault free mode

Fig 1a shows the working of the RAID 5. DISK0, DISK2, DISK3 and DISK4 are the non-faulty disk and DISK1 is a faulty disk. The RAID controller accesses the surviving disks to fetch all data and parity in the same stripe to regenerate the lost data. Therefore, to service one cache miss, several read requests are needed depending on the RAID organization. Fig 1b shows the working of RAID 5 in faulty mode. The RAID controller accesses the surviving disks to fetch all data and parity in the same stripe to regenerate the lost data. Therefore, to service one cache miss, several read requests are needed depending on the RAID organization.

Furthermore, considering the critical cases in disk arrays tolerating two or more concurrent failures, such as RAID-6, with the increased number of faulty disks, the misses of requests to faulty disks grow accordingly. Thus, the disk arrays would suffer from much more additional requests than that under the case of single disk failure. As a result, the reliability and the performance problems would be more significant. Therefore, in parity-based disk arrays under faulty conditions, a miss to faulty disks is much more expensive than a miss to other non-faulty disks.
The three contributions are,
1. VDF
2. VDF-LRU
3. VDF-LFU

A. VDF

Victim **D**isk **F**irst

The existing cache algorithms treat the blocks of faulty and non-faulty disks in the same manner so the cost of access to the faulty disk is much higher than the blocks of non-faulty disks, which is because of the more requests directed to the non-faulty disks due to the access of the faulty disk. VDF [8][13] cache is employed which aims in Obtaining the reliability and to speed up the reconstruction duration. Once a miss occurs, typically a block should be evicted from cache, and the missed block would be loaded to the free space. General replacement algorithms evict the block with the smallest access

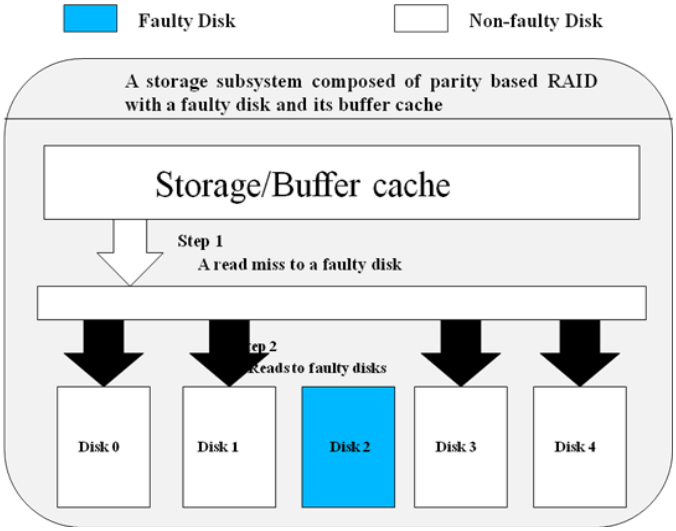

Fig 1b: RAID 5 working in faulty mode

Probability (pi) to reduce the total access probability of the remaining blocks out of buffer cache.

VDF evict the block by considering the miss penalty (Mpi) along with the access probability of the block. When the block resides in cache its Mpi is,

$$Mpi = 0$$

When the block is evicted from the cache its miss penalty depends on the location of the block i.e. either in faulty (or) non-faulty disks. The eviction of the block should be based on

$$Pi*Mpi$$

The block with the minimum access probability and miss penalty should be chosen for replacement while applying the cache replacement algorithms.

B. VDF-LRU

The cache replacement algorithm applied along with the VDF is VDF-LRU. The cache replacement algorithm LRU replaces the blocks of the cache memory when chosen for replacement. A counter is maintained in each blocks of the cache. The counter is incremented whenever the block is accessed. The block with the low counter value is replaced with the newly requested block. The VDF-LRU[15] does not choose the blocks of the faulty disk. The blocks of the faulty disk will be maintained in the cache itself which reduces the miss penalty of the faulty disk and reduces the number of requests directed to the faulty disk which is caused by accessing the faulty disk.

C. VDF-LFU

The cache replacement algorithm applied along with the VDF is VDF-LFU. The cache replacement algorithm LFU replaces the blocks of the cache memory when chosen for replacement. A counter is maintained in each blocks of the cache. The counter is incremented

whenever the block is accessed. The block with the low counter value is replaced with the newly requested block.

The VDF-LFU does not choose the blocks of the faulty disk. The blocks of the faulty disk will be maintained in the cache itself which reduces the miss penalty of the faulty disk and reduces the number of requests directed to the faulty disk which is caused by accessing the faulty disk.

III. IMPLEMENTATION

The cache memory is the smallest memory. In this project the cache contain the data from the RAID disk. The cache memory is divided into blocks. Fig 2 shows the structure of cache Memory. The structure contains the Validbit, tag and data.

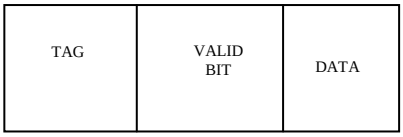

Fig 2: Structure of cache Memory

A.TAG

The tag contains the address of the data block that is fetched from the RAID disk.

B.DATA BLOCK:

It contains the actual data fetched from the RAID disk.

C.VALIDBIT:

It is used to indicate whether the data is present in that cache location.

Here the cache structure is implemented using array. The cache contains the data from the RAID disk. The tag which is the address of the data is assumed to have the disk's number and the corresponding disk's block number.

TAG= (d, b)

The Validbit is initially set to 0 to all the blocks in the cache. When a cache entry is made the particular block number is changed to 1. Fig 3 shows the cache replacement of blocks.

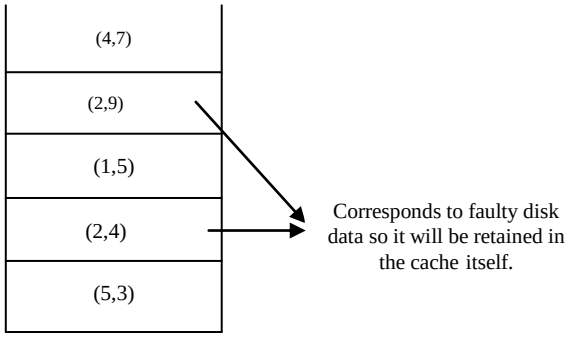

Fig 3: Cache Replacement of blocks

1)PSEUDO CODE

VDF-LRU

Replacing xi using VDF-LRU

Steps:

1. Consider the input x1, x2, x3, …, xn ,…
   where xi =address of the cache data blocks.

xi's Format =(Disk number, block number).

2. If xi is in the Local stack, when cache hit occurs, record the access timestamp and update it to the timestamp of the currently accessed block.

3. Move xi to the top of the Local stack and the global stack.

4. If xi is in the currently accessed block then update the currently accessed block by 1.

5. When a cache miss has occurred and if the cache is full, then the block at the bottom of the stack is chosen for replacement.

6. If the Local stack corresponds to the faulty disk then its weight is calculated as,

(GTS-TS)*(n-1)

7. Delete the block with the maximum weight to get the free block.

8. If the cache is free then, get the FREE block load xi to the free block; update the access timestamp of the block to the timestamp of the currently accessed block.

9. Add xi to the head of both the Local and the global stack.

2) SIMULATION OF CACHE

The Cache memory contains the data from RAID level 5. The data blocks from faulty and Non-faulty disk data are stored in cache on request. The Tag field in cache contains the disk number and the block number of that disk.

Tag	Validbit
03	1
49	1
26	1
37	1
18	1
27	1
04	1

Table 1: Values in cache Memory

E.g. consider the address value 03 from Table 1
0 corresponds to the **disk number** in RAID 5 system.
3 correspond to the **block number in the disk 0** in RAID 5 system.

3)SIMULATION OF RAID DISK

The RAID level 5 disk is simulated to store the data. The RAID level 5 is assumed to have both faulty and non-faulty disk. The RAID disk is simulated using array structure. The Raid is simulated to have 5 disks. The Table 2, 3, 4 represents the data of different disk in RAID level 5.

Address	Data
0	6
1	8
2	2
3	7

Table 2 RAID disk 0

Address	Data
0	5
1	3
2	1
3	9

Table 3 RAID disk 1

Address	Data
0	0
1	1
2	6
3	2

Table 3 RAID disk 2(Faulty as assumed)

4)VDF-LRU

The VDF-LRU cache replacement algorithm is applied to replace the blocks of the non-faulty disk when cache memory is full. The VDF-LRU uses the stack structure when applying the replacement algorithm. The VDF-LRU takes Tag value i.e. the disk number and block number as input and replaces the block at the bottom of the stack if it does not correspond to the faulty disk.

Stack containing the address of data block. Here the disk 2 is assumed to be faulty .So while replacing, **disk 2** data block is not chosen for replacement. When a new request is made for the data block already in cache, it is moved to the top of the stack.

TOP

[46, 04, 27, 18, 37, 26, 49, 04]

E.g.1 when a new request is to the block **18.**The data block 18 is already in stack (hit), so the data block **18** will be moved to the top of the stack indicate that it is.

TOP

[46, 04, 27, 37, 26, 49, 04, 18]

E.g. 2 Consider a new request to the data block **35** which is not in cache **(Cache Miss).** The Least accessed block **46** is selected for replacement and 46 is removed from the stack and the new request block **35** is added to the top of the stack.

TOP

[04, 27, 37, 26, 49, 18, 35]

E.g.3 Again when a new request is made to the data block **40** then the least accessed value 04 in cache is selected for replacement and is removed from the stack and the new data block **40** is added to the top of the stack.

TOP

[27, 37, 26, 49, 18, 35, 40]

E.g. 4 Now again consider the new request to the data block 15. Here the disk **2** corresponds to the **faulty disk** as assumed, hence cache block **27** is not selected for replacement and is maintained in the cache itself, the next least accessed data block from the stack that corresponds to the non-faulty disk, so the data block **37** is removed from the stack and **15** is added to the top of the stack.

TOP

[27, 26, 49, 18, 35, 40, 15]

In case of LRU, the least accessed block 27 is chosen for replacement even though **disk 2** is **faulty** and is removed from stack.

IV. PERFORMANCE EVALUATION

The performance analysis of **LRU** and **VDF-LRU** is compared. The LRU replaces the block at the both of stack considering the faulty and non-faulty data blocks in the same manner. The VDF-LRU treats the blocks of faulty disk more favorable and the faulty disk blocks are retained in cache memory itself.

A.LRU

LRU replacing the blocks of the data block with the minimum access values considering the faulty and non-faulty disk in the same manner and hence when a faulty disk is requested all the other non-faulty disks are accessed. Hence the miss penalty of accessing the faulty disk is higher. Consider the table 4 as an Input to LRU. Fig 4 shows the Miss penalty of accessing the blocks to the cache. Miss penalty is the number of disk accesses made. The Miss Penalty of Cache Hit is "0". In case of Cache Miss, if it corresponds to faulty disk, its Miss Penalty depends on the number of disk (in this project it is 5). If it corresponds to Non-faulty disk, the Miss penalty is 1.

B.VDF –LRU

VDF-LRU retains the blocks of faulty disk in cache memory. Hence the cost of access to the faulty disk is reduced. Consider the input in Table 5. Figure 5 shows the Miss Penalty of accessing the blocks in the cache.

C.COMPARING THE PERFORMANCE

Comparing the Miss penalty i.e. the number of request of the LRU and VDF-LRU. Figure 6 shows the performance of VDF-LRU is better compared to LRU. Since LRU replaces the recently accessed data irrespective of faulty and non-faulty blocks.

Table 6 shows the compared results of Miss Penalty (i.e. number of disk access) with 10 requests, 50 requests and 100 requests using cache replacement algorithms namely LRU and VDF-LRU. Fig 7 shows the graphical representation of the comparison.

Request number	Data Block Address	Cases
1	36	Miss
2	42	Hit
3	22	Miss
4	18	Miss
5	54	Miss
6	25	Hit
7	38	Hit
8	23	Miss
9	48	Miss

Table 5: Input to LRU

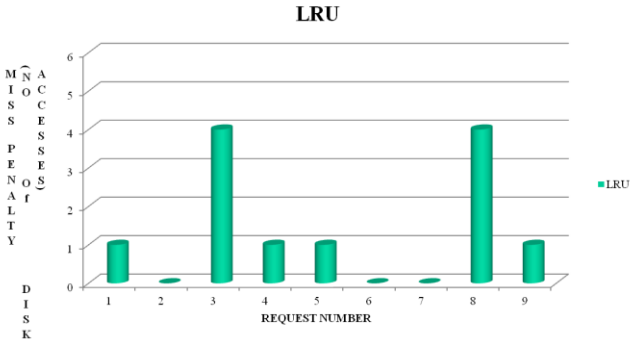

Fig 4: Analysis Result of LRU

Request number	Data Block Address	Cases
1	36	Miss
2	42	Hit
3	22	Hit
4	18	Miss
5	54	Miss
6	25	Hit
7	38	Hit
8	23	Hit
9	48	Miss

Table 6: Input to VDF-LRU

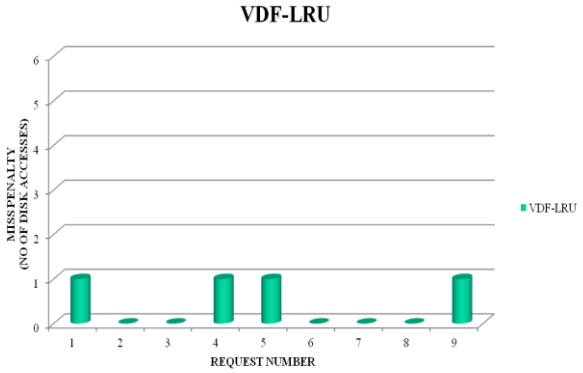

Fig 5: Analysis result of VDF-LRU

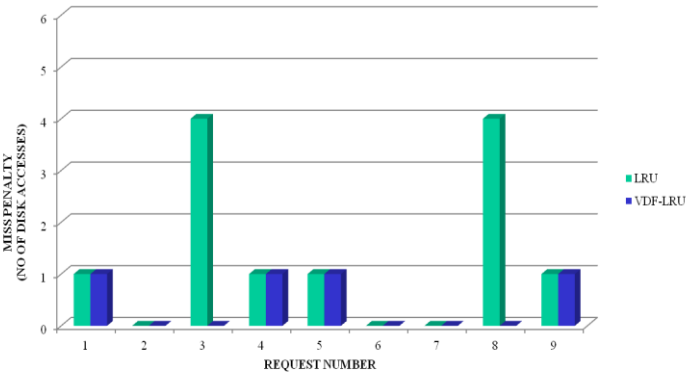

Fig 6: Comparing the analysis results of LRU and VDF-LRU

Number of Requests	Miss Penalty of LRU	Miss Penalty of VDF-LRU
10	15	4
50	675	189
100	1254	376

Table 7: Comparing the miss penalty of LRU and VDF-LRU

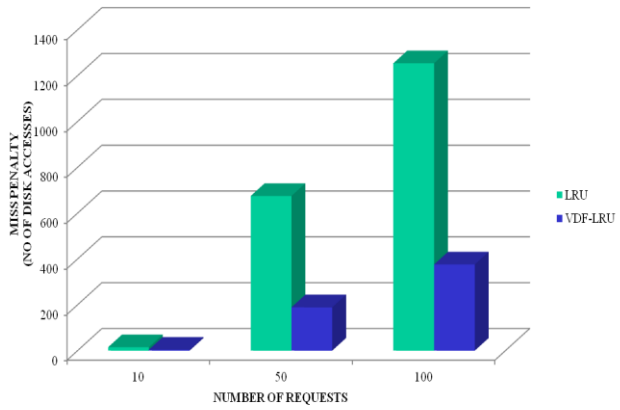

Fig 7: Comparing the Miss Penalty of LRU and VDF-LRU

V. Conclusion

In order to reduce the number of requests made to non-faulty disks of the cache, when one disk in the disk array fails, efficient cache utilization is implemented. A penalty-aware buffer cache replacement strategy, named VDF cache, to improve the reliability and performance of a RAID-based storage system. The basic idea of VDF is to treat the faulty disks more favorably, or give a higher priority to cache the data associated with the faulty disks. The benefit of this scheme is to reduce number of the cache miss directed to the faulty disk, and thus to reduce the I/O requests to the

non-faulty disks to improve the performance of the disk array.

VI. Future Work

In future, the other cache replacement algorithms like **F**irst **in First out** (FIFO) and Random replacement strategy can be applied along with VDF to analysis the performance. The VDF cache essentially reduces the requests to the non-faulty disks, it can be integrated into other faulty disk mode like data, parity, spare layout and reconstruction workloads.

References

[1] Shenggang Wan, Xubin He, Senior Member, IEEE, Jianzhong Huang,Qiang Cao, Member, IEEE, Shiyi Li, and Changsheng Xie, Member, IEEE. *"An Efficient Penalty-Aware Cache to Improve the Performance of Parity-Based Disk Arraysunder Faulty Conditions"* IEEE Transactions On Parallel And Distributed Systems, Vol. 24, No. 8, August 2013.

[2] S. Wu, H. Jiang, D. Feng, L. Tian, and B. Mao, "*WorkOut: I/O Workload Outsourcing for Boosting RAID Reconstruction Performance*,"Proc. Seventh USENIX FAST Conf., 2009.

[3] G.K.M, X. Li, and J.J. Wylie, "*Flat XOR-Based Erasure Codes in Storage Systems: Constructions, Efficient Recovery, and Tradeoffs*," Proc. IEEE 26th Mass Storage Systems and Technologies Conf. (MSST), 2010.

[4] S. Wan, Q. Cao, C.S. Xie, B. Eckart, and X. He, "*Code-M: A Non-MDS Erasure Code Scheme to Support Fast Recovery from up to Two-Disk Failures in Storage Systems*," Proc. IEEE/IFIP DSN Int'l Conf. Dependable Systems and Networks, 2010.

[5] Y. Cassuto and J. Bruck, *"Cyclic Lowest Density MDS Array Codes,"* IEEE Trans. Information Theory, vol. 55, no. 4 pp. 1721- 1729, Apr. 2009.

[6] C. Jin, H. Jiang, D. Feng, and L. Tian, "*P-Code: A New RAID-6 Code with Optimal Properties"* Proc. 23rd Int'l Conf. Supercomputing (ICS '09), pp. 360-369, June 2009.

[7] C. Wu, X. He, G. Wu, S. Wan, X. Liu, Q. Cao, and C. Xie, *"HDP Code: A Horizontal-Diagonal Parity Code to Optimize I/O Load Balancing in Raid-6,"* Proc. IEEE 41th Int'l Conf. Dependable Systems and Networks, June 2011.

[8] S. Wan, Q. Cao, J. Huang, S. Li, X. Li, S. Zhan, C. Xie, L. Yu, and X. He*, "Victim Disk First: An Asymmetric Cache to Boost the Performance of Disk Arrays under Faulty Conditions,"* Proc. USENIX Ann. Technical Conf., June 2011.

[9] *"Raid6 Definition in Dictionary r.,"* SNIA, http://www.snia.org/education/dictionary/r, 2007.

[10P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar, "*Row-Diagonal Parity for Double Disk Failure Correction,"* Proc. Third USENIX FAST Conf. File and Storage Technologies, pp. 1-14, Mar. 2004.

[11] C. Wu, X. He, G. Wu, S. Wan, X. Liu, Q. Cao, and C. Xie, "*HDP Code: A Horizontal-Diagonal Parity Code to Optimize I/O Load Balancing in Raid-6,"* Proc. IEEE 41th Int'l Conf. Dependable Systems and Networks, June 2011.

[12] S. Kavalanekar, B. Worthington, Q. Zhang, and V. Sharda, *"Characterization of Storage Workload Traces from Production Windows Servers,"* Proc. IEEE Int'l Symp. Workload Characterization, 2008.

[13] S. Wan, Q. Cao, J. Huang, S. Li, X. Li, S. Zhan, C. Xie, L. Yu, and X. He*, "Victim Disk First: An Asymmetric Cache to Boost the Performance of Disk Arrays under Faulty Conditions,"* Proc. USENIX Ann. Technical Conf., June 2011.

[14] S. Jiang, X. Ding, F. Chen, E. Tan, and X. Zhang, *"DULO: An Effective Buffer Cache Management Scheme to Exploit Both Temporal and Spatial Localities,"* Proc. Fourth USENIX FAST Conf., 2005.

[15] S. Jiang and X. Zhang, *"Making LRU Friendly to Weak Locality Workloads: A Novel Replacement Algorithm to Improve Buffer Cache Performance,"* IEEE Trans. Computers, vol. 54, no. 8, pp. 939- 952, Aug. 2005.