International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 4 Issue: 5 192 - 194

Enhancement of Writeback Caching by changes in flush and its parameters

Chandan J
M. Tech in Software Engineering
Dept of ISE, RV College of Engineering
Bengaluru, India
chandanjayaram182@gmail.com

Kavitha S N
Assistant Professor
Dept of ISE, RV College of Engineering
Bengaluru, India
kavithasn@rvce.edu.in

***Abstract*—** Achievement of high performance in computing or accessing of data is the aim of any system. Reduction of access time to a particular data which is present in the device is very important for the enhancement in the performance. Caching is implemented to do the same. The group of cache device and the virtual device is made as a cache group to enhance the performance of the system. The system may not be on the same condition different instances of time. There will always be a variation in io rates of the application, which is not utilized for the full extent. These differences in the io rates can be utilized effectively for the enhancement of the performance of the system. When the system is idle of with less io then the system will force the flush so that the inconsistency of data is reduced. When the system is being bombarded with io then less threads are given for the flush io. These variations in the threads assigned for the implementation of flush io will enhance the overall performance of the system.

***Keywords*—** *CG-Cache group, CD-cache device, VD-virtual drive, Dirty data, Valid data, cache hit, cache miss.*

*****

I. INTRODUCTION

In any high end system on which there are lots and lots of io which is executed per second. These ios has to be executed efficiently and the same has to be acknowledged back to the application which has initiated the io. The caching is very important for any high end system. When the application is write intensive then the write back method of caching is used. The major drawback of this method is the inconsistency of data in the system at certain times. Since a cache device has to have a less access time, SSDs are used for CD. Rotational media can be used for VD. The combination of VD and CD is a CG on which the ios has to be triggered. Frequently accessed data is stored in the cache device. This data is detected by hot detection algorithm. Any chunk of data which is requested by the application more than 3 times is popped on to the CD. Then after some time depending on the system the data is flushed to maintain the data consistency.

II. OVERVIEW

A. *Reasearch gap*

In the previous implementation of write back caching solution there was a single caching algorithm which was running in background. The problem with this method was the threads which are utilized for the flush io use to be idle when no data was dirty and the same were overloaded when there used to be too much of dirty data to be flushed on to the virtual device.

The problem of with this method was leading the system to achieve less performance. This can be eliminated by variation of flush parameters according to the variations in flush.

B. *Methodlogy*

The solution is designed such that the flush parameters will vary according to the variations in the application's io. The solution has been designed to have 3 variations in the flush io.

- Idle flush
- Periodic flush
- Forced flush

Idle flush is triggered when there are less or non io on the CG. Periodic flush is a middle layer flush which is triggered on the device when the dirty data count exceeds 45% of the total cache size and works till it reaches 60%. When the cache device is filled with more than 60% of dirty data then the forced flush is triggered which will allocate maximum allowable threads for flush io.

III. IMPLEMENTATION LOGIC

The idea which is briefly said in the previous section is explained in the implementation logic. Here the minute details of design and the requirements are stated so that the idea is well established and understood.

Dirty data are the data present in the CD which is not sent to the VD through flush. Valid data is the super set of dirty data but also include those chunks of data which are sent to VD also. Cache hit refers to the situation in the data needed by the io is present in the cache device. Cache miss is the negative case of cache hit, where the requested data is not present in the CD. Partial hit refers to the situation in which the data chunks needed are partially present in the device. These terminologies are repeatedly used.

A. *Assumptions*

The CD is divided logically into blocks of 64KB size called as cache block. Each cache block is internally divided into cache line of 4K. So each cache block is having 16 cache lines. There will be a dirty bit for each cache line. Each 4K io is marked for dirty bit after the write is performed on the same. The metadata is present in the CD

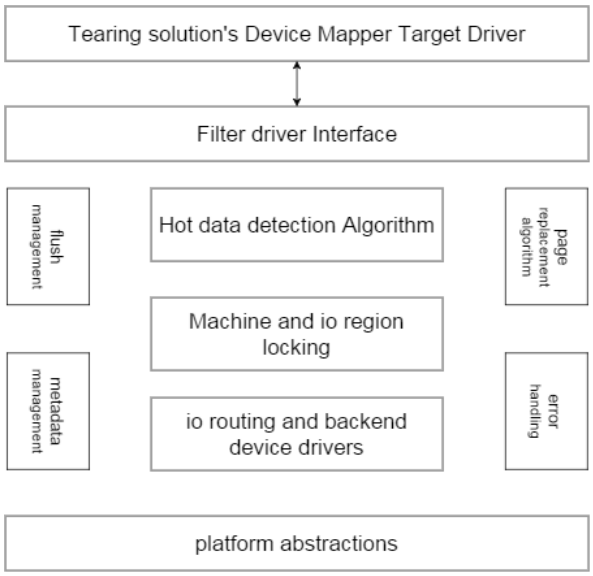
and in VD also. Configuration details are in CD which is considered primarily. In any situation of inconsistence state of system the system is brought back to the working state using the configuration details present in the CD.

Sequential ios are not serviced by the solution. The speed of the device's spindle is utilized for these sequential ios.

Ios which are greater than the size 32K are also bypassed by the solution. If there is any overlap in the ios greater than the size of 32K then the ios are serviced by the solution.

B. Io work path

When a io is encountered by the device, its first mapped to the solution's driver, where the parameters of the io are recognized. if the size of io is greater than bypass size then the virtual windows are assigned to the io. Else physical window is assigned to the io in the cache device. Then the ios are mapped to the respective LBA in the respective devices



The diagram above describes the overall working of the io in the system. Any io which is coming to the system and how it is executed in the system. The device mapper will get all the io and decide where it has to be mapped. The filter driver will check for the device has to go through the CD or VD depending on the bypass size defined. Hot data detection algorithm will serve to recognize the frequently accessed by the application. Machine region locking is done for maintaining the data integrity. Io routing is done in the back end to the exact location of the device and interacts with the device manager of the drive. Depending on the device the abstractions are done. Flush manager will kicks in when there are less ios or the conditions for the flush is met. Metadata management is very important for maintaining the configuration both CD and VD. The copy of metadata is maintained both in CD and VD. Multiple copies are maintained for the consistency of the configuration even when there is any inconsistency in the system. Error handling is done so that the some inherent errors in the commands are managed.

C. CD flush

The CD flush is for a particular CD specified by its SCSI id. Caching is disabled when the CD flush is issued. Dirty tree is always maintained for a particular VD. In a loop each VD in the CG is verified for the dirty data of that CD is flushed. Dirty tree is an AVL tree after each deletion of the node its balanced. CD flush is a trouble process. During the cache disable any partial or full hit on the CD is come across then its serviced.

D. VD flush

VD is flush is simple to implement. In any CG the data pertaining to the VD specified by its SCSI id is flushed. There will be no cache disable during the VD flush. The dirty tree as a whole is flushed to the device. Its better to stop the io on the VD before the VD flush, else from one end the device will be clearing the dirty tree and from the other side it will be building.

E. Flush parameters

Flush io is characterized by 3 parameters.

- Rate of flush
- Time interval between flush
- Size of flush

These 3 parameters can be varied even by the user but only during the period of 45% to 60% of the total cache size. The parameters for the remaining types of flush are fixed.

ACKNOWLEDGMENT

Any accomplishment is not possible without the help of people who supported the endeavor in all sorts and encouraged the work which was carried out. The department of ISE, RVCE for all their support in all the facilities and the encouragement provided in carrying the work. Thanking friends and family is very important in this course of work without which it would be impossible even in carrying the work finishing it in due time.

REFERENCES

[1] N. P. Jouppi, "Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers," in Proc. Int. Symp. on Computer Architecture, 1990, pp. 364–373.

[2] ——, "Cache Write Policies and Performance," in Proc. Int. Symp. on Computer Architecture, 1993, pp. 191–201.

[3] P. P. Chu and R. Gottipati, "Write Buffer Design for On-Chip Cache," in Proc. IEEE Int. Conf. on Computer Design, 1994, pp. 311–316.

[4] H.-H. S. Lee, G. S. Tyson, and M. K. Farrens, "Eager Writeback - A Technique for Improving Bandwidth Utilization," in Proc. ACM/IEEE Int. Symp. on Microarchitecture, 2000, pp. 11–21.

[5] S. P. E. Corporation, http://www.spec.org/.

[6] T. Austin et al., "Simplescalar 3.0," http://www.simplescalar.com/.

[7] M. K. McKusick, W. N. Joy, S. J. Leffler, and R. S. Fabry. A fast file system for unix. ACM Trans. Comput. Syst., 2(3):181–197, Aug. 1984.

[8] N. Megiddo and D. Modha. ARC: A self-tuning, low overhead replacement cache. In Proc. of USENIX FAST, pages 115–130, 2003.

[9] D. Narayanan, A. Donnelly, and A. Rowstron. Write Off-Loading: Practical Power Management for Enterprise Storage. In Proc. of USENIX FAST, 2008.

[10] C. Ruemmler and J. Wilkes. An Introduction to Disk Drive Modeling. Computer, 2:17–28, 1994.
[11] R. Koller and R. Rangaswami. I/O Deduplication: Utilizing Content Similarity to Improve I/O Perfor-mance. In Proc. of USENIX FAST, pages 211–224, February 2010.
[12] N. Megiddo and D. Modha. ARC: A self-tuning, low overhead replacement cache. In Proc. of USENIX FAST, pages 115–130, 2003
[13] M. K. Qureshi, M. M. Franceschini, and L. A. Lastras-montao. Improving read performance of phase change mem-ories via write cancellation and write pausing. In Pro-ceedings of the 2010 IEEE 13th International Symposium on High Performance Computer Architecture, pages 1–11, 2010
[14] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens. Memory access scheduling. In Proceedings of the 27th annual international symposium on Computer architec-ture, ISCA '00, pages 128–138, New York, NY, USA, 2000. ACM
[15] S. Srinath, O. Mutlu, H. Kim, and Y. N. Patt. Feed-back directed prefetching: Improving the performance and bandwidth-efficiency of hardware prefetchers. In High Per-formance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on, pages 63–74, 2007.