COSMA: An Efficient Concurrency-Oriented Space Management Scheme for In-memory File Systems

Chunhua Xiao*, Zipei Feng*, Ting Wu*, Lin Zhang*, XiaoXiang Fu*, WeiChen Liu†
*College of Computer Science, Chongqing University, Chongqing, China
†School of Computer Science and Engineering, Nanyang Technological University, Singapore

***Abstract*—Emerging file systems have been designed for fully exploring NVM's advanced features. However, with the development of big data, these file systems suffer from the performance degradation in highly concurrent environment, which are caused by serious access conflicts in space management. To solve this problem, we propose an efficient concurrency-oriented space management scheme named as COSMA. Along with novel data structure design, COSMA is able to greatly reduce request congestion among multiple threads through hierarchical space allocation scheme. Furthermore, COSMA provides 3 reclamation strategies to improve space utilization, and can also adapt to different systems which varied in NVM capacities. To ensure the system reliability, COSMA is capable of keeping wear leveling among multiple NVMs slots. We implement COSMA in a representative persistent file system, PMFS. Experimental results show that COSMA can improve the IOPS of PMFS by 15%, the write throughput of PMFS by 7.6% and the concurrent processing performance of PMFS by 50%. Besides, it can also achieve wear-leveling among multiple NVMs.**

***Index Terms*—NVM, File system, Concurrency, Wear-leveling**

I. Introduction

Emerging byte-addressable non-volatile memory (NVM), such as PCM [1] and Intel 3D XPoint [2], has the advantages of fast, cheap and persistent, and is considered as the next generation of persistent memory. Many file systems have been designed for NVM, such as PMFS [3], NOVA [4] and EXT4-DAX [5]. Unfortunately, these file systems ignore the space allocation performance in multi-threading and highly concurrent environments, the performance of the space allocator becomes a bottleneck. For example, PMFS uses a global in-use list to manage NVM space, it locks space allocator in each space allocation and reclamation. NOVA distributes NVM space equally to all CPU cores, each core manages a range of space exclusively, but it has no effect in systems with few CPUs.

To solve the problem of performance degradation, this paper proposes a file system space management scheme named as COSMA, which is effective for high concurrency processing. COSMA provides both public and thread-based allocation areas to reduce competition for space allocator among threads, so that its concurrency performance can scale well in various system. For systems with different NVM capacity, COSMA offers various strategies for space reclamation: 1) RPFSC. 2) T_LRU. 3) DMDL_LRU. Moreover, COSMA supports systems with multiple NVMs and provides wear-leveling among multiple NVMs. We implemented COSMA in PMFS, the experimental results show that COSMA can improve the IOPS of PMFS by 15%, the write throughput of PMFS by 7.6% and the concurrent processing performance of PMFS by 50%.

In summary, this paper makes the following contributions:

- We prove that the existing NVM file systems have obvious performance degradation as much as 500% in highly concurrent environment.
- We propose a concurrency friendly space management scheme COSMA. 1) It provides a hierarchical space management scheme to reduce competition for space allocation among threads. 2) According to the NVM capacity of different systems, we proposed 3 efficient space reclamation strategies to optimize its space utilization. 3) To improve reliability of NVM, COSMA supports both load-balance and wear-leveling among multiple NVMs.
- We implemented COSMA in PMFS, the experimental results show that COSMA can improve the IOPS by 15%, the write throughput by 7.6% and the concurrent processing performance by 50%.

The remainder of this paper is organized as follows. Section II introduces the background and motivation. Section III presents the design of COSMA. Section IV presents the evaluation results. This paper is concluded in Section V.

II. Background and Motivation

To benchmark the performance of NVM file systems, including PMFS, NOVA and EXT4-DAX in highly concurrent environment, we created many threads to writes 800KB of data to separate files. The configuration is listed in the 4th section. As shown in Fig.1, the baseline is the ideal growth curve when the performance scales perfectly. The completion time is 5 times the baseline in the worst case. For all three, the completion time shows an exponential growth trend.

Most NVM file systems manage space with a centralized space allocator, which is inefficient. As shown in Fig.2, many threads compete for one space allocator, causing severe performance loss and delay. To solve the problem. Firstly, the flat space management scheme should be changed. Secondly, it is vital to improve NVM space utilization. Thirdly, we meed improve the reliability and performance of NVMs in multiple NVM architecture through wear leveling and load balance.

Corresponding author: Chunhua Xiao is with Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, China, and College of Computer Science, Chongqing University, Chongqing 400044, P. R. China.

2576-6996/20/$31.00 ©2020 IEEE
DOI 10.1109/ICCD50377.2020.00040

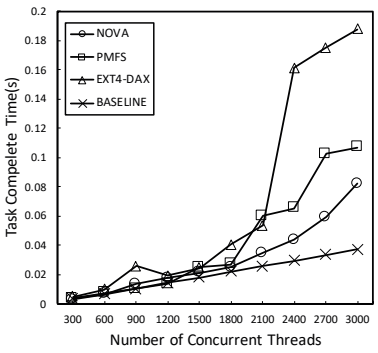


Fig. 1. Changes of task completion time in highly concurrent environment

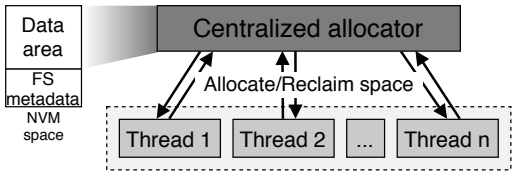


Fig. 2. Flat space management scheme

As the storage density of NVM is low, the capacity of a single NVM is limited to 512GB [6], multiple NVMs are needed to meet capacity requirements. However, the existing NVM wear leveling technologies only solve the wear leveling of a single NVM. It is necessary to solve the problem of wear leveling among multiple NVMs to avoid its lifetime decline, performance degradation and low utilization rate.

TABLE I
COMPARISON OF COSMA AND OTHER NVM FILE SYSTEMS

Space Management Scheme in Filesystems	Wear-leveling among NVMs	Performance Scalability	High Concurrency Optimized	
			Private Allocation Area	Lock Granularity
EXT4-DAX	×	×	×	File system
PMFS	×	×	×	File system
NOVA	×	×	×	CPU
COSMA	✓	✓	✓	Thread

As shown in table I, we compare COSMA with 3 typical NVM file systems: 1) COSMA provides the wear-leveling among multiple NVMs. 2) COSMA optimizes for highly concurrent environment, it can achieve better concurrency performance and can scales well in various systems. 3) COSMA has smaller granularity in the lock.

III. DESIGN

A. Overview

Fig.3 shows the structure of COSMA. We allocate a private allocation area for each thread. Since each thread allocates and reclaims space from its private allocation area, there will be no competition among threads and achieve better concurrent processing performance. COSMA divides the space of the file system into the private allocation area and the public allocation area. When a thread's private allocation space is insufficient,

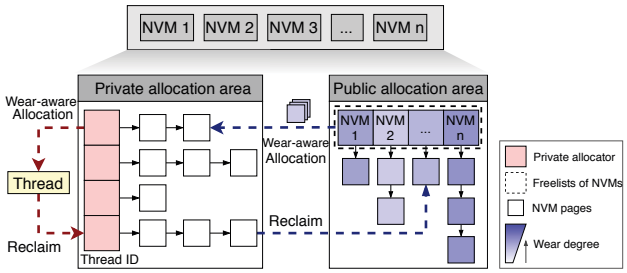


Fig. 3. COSMA space management structure

it requests space from the public allocation area. When public allocation area is insufficient, it reclaim space from threads.

B. Hierarchical Space Management Scheme

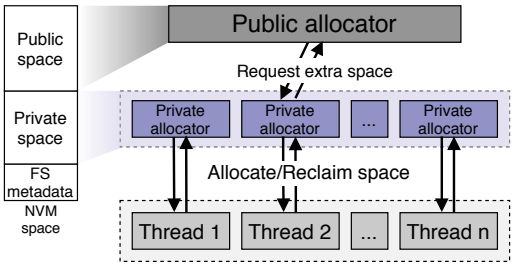


Fig. 4. Hierarchical space management scheme

We propose a hierarchical space allocation scheme shown in Fig.4, it divides the space allocator into thread local allocators and file system public space allocators, and processing requests at the local level. Each thread allocates and reclaims space from its private allocation area, which reduce the competition of threads. For example, it is only necessary for thread 2 to request for more space from the public allocation area when its private allocation area is insufficient. For other threads, all requests can be completed at local level.

C. Efficient Space Reclamation Strategy

To efficiently reclaim space, we adopt a lazy reclamation strategy, we only passively reclaim space from threads when the public allocation area is insufficient. How to select threads for space reclamation is critical, which might causes frequent reclamations and affect the performance.

1) Reclamation Policy Based on Free Space Capacity(RPFSC): RPFSC reclaims space from threads with most free space, which is suitable for systems with insufficient NVM space. The advantage is that there is no extra overhead in space allocation. However, the selected threads might be hot threads which are performing a lot of operations.

2) Timestamp Based LRU strategy(T_LRU): T_LRU select the thread that least recently allocates and reclaim space. Only the timestamp needs to be updated in allocation, and the overhead is negligible. However, when the public allocation area needs to reclaim space, the cost is relatively high. It performs well in computer systems with sufficient NVM space, but not in these with insufficient NVM space.

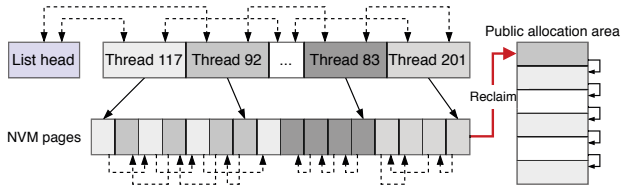


Fig. 5. DMDL_LRU

3) LRU Strategy Based on Direct Mapping and Double Linked List(DMDL_LRU): Considering the operation system limits the number of threads (45284 generally), as shown in Fig.5, we designed a LRU algorithm based on direct mapping and double linked list. The time complexity of this scheme is O(1). Each LRU update operation needs to lock the corresponding node and its front and back nodes. The cost of maintaining LRU is the same in allocation and reclamation. This scheme is less affected by the NVM capacity, and its performance is stable, which is suitable for various systems.

TABLE II
COMPARISON OF 3 RECLAMATION STRATEGIES

Strategy	Allocation Time Complexity	Reclamation Time Complexity	Space Complexity	Insufficient Space	Sufficient Space
RPFSC	O(1)	O(N)	O(N)	√	
T_LRU	O(1)	O(N)	O(N)		√
DMDL_LRU	O(1)	O(1)	O(N)	√	√

4) Comparison of 3 reclamation strategies: As shown in table II, we compare these strategies as follows. 1) RPFSC applies to systems with insufficient NVM capacity. It reclaims space from threads that occupy most resources as much as possible, thus reducing reclamation frequency. 2) T_LRU is suitable for systems with sufficient NVM capacity. It can achieve good performance, as reclamation rarely occurs and its maintenance cost is negligible. 3) DMDL_LRU applies to common situations. It has O(1) time complexity for both allocation and reclamation, which can achieve stable performance in most cases.

D. Wear leveling scheme

In COSMA, wear leveling within each NVM is implemented by hardware technology, it adapts the following policies to realize wear leveling and load balance among NVMs.

- The private allocation area choose the NVM with lowest number of writes to complete the allocation request.
- The public allocation area choose the NVM with lowest wear degree to expand the private allocation area.
- In the private and public allocation area, the NVM space is managed with several separate freelists according to NVM slots to efficiently allocate and reclaim.

In a word, we realize wear balance and load leveling among NVMs through 2 layers structure of wear leveling.

IV. EVALUATION

To verify COSMA, we conducted some experiments. We implemented COSMA on PMFS in Linux kernel 4.4.4, the most typical and widely used NVM file system. It can also be extended to other file systems.

A. Configuration of Experimental Environment

The experiments are conducted on a 4-Core platform with Intel Core i5-6500 CPU, running at 3.20 GHz, and system is equipped with 40GB memory. Since NVM is currently unavailable, we use NVM library [7] to reserve 8GB DRAM to simulate NVM devices (/dev/pmem0).

B. Performance Test

1) Microbenchmarks: COSMA is aimed at optimizing the space allocation of the file system, which will only affect writing, but not affect reading. We used Fio [8] to test the difference of write performance between the optimized PMFS and PMFS in different thread numbers and I/O size.

Fig.6(a)(b)(c) shows the comparison of sequential write throughput between COSMA and PMFS in 4, 8, 16 threads respectively. We can see that the optimization scheme we propose can improve the sequential write performance of PMFS. Among them, when IO size is 128KB, the write performance is improved most obviously compared with PMFS, which can improve the sequential write performance up to 1300MB/s, with the performance improvement ratio reaching 7.6%.

2) Macrobenchmarks: To explore COSMA in actual application scenarios, we used Filebench [9] to measure IOPS performance in workloads with parameters shown in Table III.

TABLE III
PARAMETER SETTINGS FOR FILEBENCH WORKLOADS

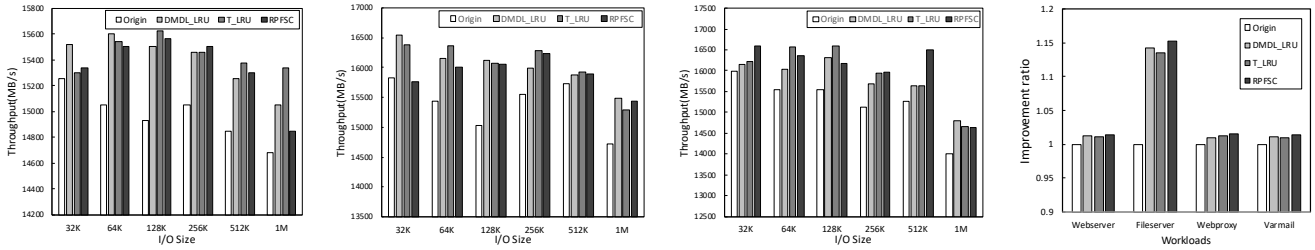
Workload	***Average file size***	***I/O size (r/w)***	***Threads***
Webserver	128 KB	1MB/4KB	50
Fileserver	128KB	1MB/4KB	50
Webproxy	16KB	4KB/4KB	20
Varmail	16KB	1MB/4KB	50

Fig.6(d) shows the performance improvement ratio of COSMA compared to PMFS in 4 workloads. We can see that COSMA has the most obvious improvement in the workload of fileserver, reaching a 15% improvement percentage, it has slightly improved in other workloads.

The reason the fileserver workload can get such improvement is that most of its operation is write, and COSMA is most obvious for scenes with more write operations. And most operations in other workloads are read.

3) Performance scalability test: In order to test the performance scalability of COSMA in a highly concurrent environment, we repeated the experiment in section 1.

Fig.7 shows the comparison of the task complete time between the COSMA and PMFS as the number of concurrent threads increases. We can see that the 3 space policies take less time to complete all tasks than PMFS. In addition, as the number of concurrent write threads increases, the gap between the



(a) 4 threads sequential write (b) 8 threads sequential write (c) 16 threads sequential write (d) improvment in 4 workloads

Fig. 6. Comparison of performance between 3 page reclamation policies and original policies

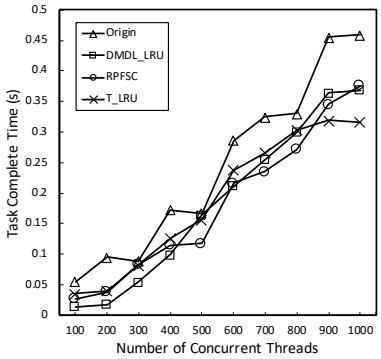


Fig. 7. Task completion time of each space reclamation policy as concurrent threads go from 100 to 1000

3 space strategies and PMFS in completing all tasks becomes larger and larger. According to the above experimental results, we can conclude that COSMA can improve the performance scalability in a highly concurrent environment.

C. Wear-leveling test

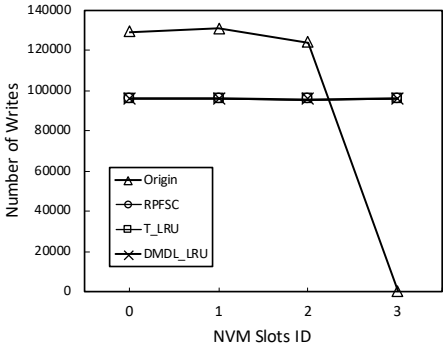


Fig. 8. Write amount of 4 NVMs under each space management policy

To test the wear-leveling effect of COSMA. We create 50 files, write 30MB of data to each file, and then obtain the write times of each NVM. Fig.8 shows the write volume of 4 NVMs between COSMA and PMFS. We can see that PMFS cannot achieve wear leveling well, and the write pressure to the NVM is concentrated on the first 3 NVMs. This will cause the first 3 NVMs to become unusable early, which reduces the reliability and availability of the file system.

V. CONCLUSIONS

In the big data era, the data generated each year increases dramatically. To efficiently processing these data, it is crucial to improve the highly concurrent performance of file systems. Existing NVM file systems have obvious performance degradation as much as 500% in a highly concurrent environment. To solve this problem, we propose a novel space management scheme called COSMA, it changes the flat space management scheme into a hierarchical one, achieving good performance scalability. Besides, it can adapt to systems with different NVM capacities and realize load balance and wear leveling among NVMs. We implement COSMA in PMFS, the experimental results show that COSMA can improve the IOPS by 15%, the write throughput by 7.6% and the concurrent processing performance by 50%.

ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China(No.61772094), Chongqing Municipal Natural Science Foundation(No.cstc2020jcyj-msxmx0724), Venture&Innovation Support Program for Chongqing Overseas Returnees(cx2019094), and the Fundamental Research Funds for the Central Universities, (No.2020cdjqy-a019, No.2020cdjlhzz-054, No.2019cdxyjsj0021).

REFERENCES

[1] Burr G W, Brightsky M J, Sebastian A, et al. Recent Progress in Phase-Change Memory Technology. IEEE Journal on Emerging & Selected Topics in Circuits & Systems, 2016, 6(2):146-162.
[2] Intel Optane Technology. https://www.intel.com/content/www/us/en/architecture-and-technology/optane-memory.html.
[3] Dulloor S, Sanjay K, Anil K, Philip L, Dheeraj R, Rajesh S, Jeff J. System software for persistent memory. Proceedings of the Ninth European Conference on Computer Systems, 2004: 1-15.
[4] Xu J, Steven S. NOVA: A log-structured file system for hybrid volatile/non-volatile main memories. 14th USENIX Conference on File and Storage Technologies, 2016: 323-338.
[5] Wilcox, Add support for nv-dimms to ext4. https://lwn.net/Articles/613384/.
[6] "Three Reasons Top Cloud Service Providers Race to Deploy Intel Optane DC Persistent Memory", Raejeanne Skillern, https://itpeernetwork.intel.com/hyperscale-cloud-memory/#gs.CbPFjHGu, 2018
[7] Persistent Memory Wiki. https://nvdimm.wiki.kernel.org/
[8] Axboe, Jens. Fio. https://github.com/axboe/fio.
[9] Tarasov V, Zadok E, Shepler S. Filebench: A flexible framework for file system benchmarking. USENIX, 2016, 41(1): 6-12.