FAGC: Free Space Fragmentation Aware GC Scheme based on Observations of Energy Consumption

Lihua Yang[1,2], Zhipeng Tan[1], Fang Wang[1], Yang Xiao[1], Wei Zhang[2], Biao He[3]
[1] Wuhan National Laboratory for Optoelectronics, Key Laboratory of Information Storage System
Engineering Research Center of Data Storage Systems and Technology, Ministry of Education of China
School of Computer Science and Technology, Huazhong University of Science and Technology
[2] College of Science and Technology, National University of Defense Technology, China
[3] Huawei Technology Co., Ltd, China
yanglihua@nudt.edu.cn, {tanzhipeng, wangfang, menguozi}@hust.edu.cn, weizhang@nudt.edu.cn, hebiao6@huawei.com

***Abstract*—Smartphones are everyday necessities with limited power supply. Charging a smartphone twice a day or more affects user experience. Flash friendly file system (F2FS) is a widely-used log-structured file system for smartphones. Free space fragmentation of F2FS consists of invalid blocks mainly causing performance degradation. F2FS reclaims invalid blocks by garbage collection (GC). We explore the energy consumption of GC and the effect of GC on reducing free space fragments. We observe the energy consumption of one background GC is large but its effect on reducing free space fragments is limited. These motivate us to improve the energy efficiency of GC. We reassess how much free space is a free space fragment based on data analysis, use the free space fragmentation factor to measure the degree of free space fragmentation quickly. We suggest the free space fragmentation aware GC scheme (FAGC) that optimizes the selection for victim segments and the migration for valid blocks. Experiments on real platform show that FAGC reduces GC count by 82.68% and 74.51% respectively than traditional F2FS and the latest GC optimization of F2FS, ATGC. FAGC reduces the energy consumption by 164.37 J and 100.64 J compared to traditional F2FS and ATGC respectively for a synthetic benchmark.**

I. Introduction

Smartphones provide limited battery power due to portability. Charging a smartphone twice a day or more affects user experience. In the past decade, researchers have focused on exploring the energy consumption model of a smartphone and proposing optimization schemes based on usage scenarios. Flash friendly file system (F2FS) [1] is a widely-used file system for smartphones. However, there is little research to understand its energy consumption characteristics.

When the continuous free space of file system is exhausted, there is a large amount of free space fragments. F2FS triggers garbage collection (GC) to obtain free segments or uses threaded logging write scheme to overwrite free space fragments that causes fragmentation of newly written files directly [2]. We observe the energy consumption of one background (BG) GC is relatively large and the effect of BGGC on reducing free space fragments is limited. It is better to reduce GC count and make each GC reduce free space fragments as much as possible. We recommend improving the energy efficiency of GC recycling invalid data.

This work was supported in part by NSFC (U22A2027, 61832020), Foundation of State Key Lab of HPC (202101-03), Natural Science Foundation of Hunan Province (2021JJ40692), the research project of NUDT (ZK20-09), Foundation Enhancement Project (2022-JCJQ-JJ-0318). Zhipeng Tan is the corresponding author (tanzhipeng@hust.edu.cn).

We propose a GC scheme that is aware of the degree of free space fragmentation. We reassess the size of a free space fragment based on data analysis, design the free space fragmentation **f**actor (f) to measure the degree of free space fragmentation, and propose FAGC scheme. We modify the algorithm for selecting victim segments that trades off among segment utilization, age, and free space fragmentation. FAGC migrates valid blocks using threaded logging writes. Experimental results show that the GC count of FAGC is 82.68% lower than that of traditional F2FS and 74.51% lower than that of the latest GC optimization scheme of F2FS, ATGC (Age-Threshold based Garbage Collection) [3]. Compared to traditional F2FS and ATGC, FAGC reduces the energy consumption by 164.37 J and 100.64 J for a synthetic benchmark, respectively.

II. Design of FAGC

We use our developed energy consumption measurement system based on Kirin 9000 [4] to understand the energy consumption of F2FS. The energy consumption of one BGGC cannot be ignored that the minimum energy consumption of BGGC we measured is 102.79 mJ. The battery capacity of a mid-end smartphone is 3,000 mAh and the smartphone battery voltage in industry is commonly 3.8 V. A battery can provide 41,040 J when fully charged. Assuming that the smartphone with full power can be used for 24 hours, it consumes 475 mJ per second. If adding 1,000 BGGCs per hour, the standby time is shortened by 1.36 hours. The energy consumption of GC is relatively large and the number of GCs should be reduced under the premise that system works normally. However, we observe that the effect of energy-intensive BGGC on reducing free space fragments is limited based on changes in the number of free space fragments.

Reassessment of free space fragments. How to distinguish between free space and free space fragment becomes a key issue in selecting victim segments. Since most I/O requests in smartphones are small, a relatively **large free space fragment** can satisfy most I/O requests that should be regarded as **available free space**. According to the traditional definition of free space fragmentation, only when the free space is

978-3-9819263-7-8/DATE23/© 2023 EDAA

completely contiguous is there no fragmentation. In order to ensure the continuity of free space, GC of F2FS will be aggressive, resulting in large GC overheads.

We define the free space less than 128 KB as a free space fragment. This is because (1) 128 KB is the default size for readahead in Linux kernel. Pre-read data can be placed in the free space of 128 KB and a read request can be satisfied by one I/O instead of being split into multiple I/O requests. (2) We find read and write I/Os less than or equal to 128 KB account for 85% and 87%, respectively, by analyzing applications traces.

Measurement of free space fragmentation. We design the free space fragmentation **f**actor (f) to measure the degree of free space fragmentation. It is calculated by *valid_map* in segment information table where bit 1 indicates valid data. The threshold for a free space fragment (128 KB) divided by the size of a block (4 KB), the threshold value is 32. There are X consecutive zeros in the bitmap, denoted as X (X<32). Assuming that there are n fragments in a segment, $f = \sum_{i=1}^{n} \frac{1}{X_i}$, the larger f is, the more fragmented. Since floating-point arithmetic is not recommended in the kernel, we make f an integer via $f = \sum_{i=1}^{n} \frac{1}{X_i} \times 126 \times 127$. Multiplying by 127 is to get distinct integers and enlarging by 126 is to distinguish free space fragments of different sizes quickly.

Design of FAGC. FAGC how to work is shown in Algorithm 1. The age_weight is 40, the f_weight is the same as age_weight. The larger age, f, and u, the smaller the migration cost according to $cost = UINT\_MAX - (age + f + u)$ is. UINT_MAX is the largest unsigned integer. Age, f, and u are all normalized that migration cost will not tend to be affected by a single factor. We select the segment with more serious free space fragmentation with the f of a segment. When migrating valid blocks, FAGC selects the target segment with a similar age to the victim segment and migrates valid blocks to the target segment via threaded logging writes. FAGC writes invalid or free blocks in the target segment one by one. Different from traditional F2FS and ATGC, FAGC selects segments with severe free space fragmentation. FAGC only performs GC on critical segments that greatly affect performance to reduce GCs and minimizes fragmentation per GC.

Algorithm 1 FAGC scheme

Input: max_mtime, $mtime$, $total\_time$, $vblocks$, f, max_f, min_f, $segno$, age_weight, f_weight, min_cost, $oldest\_age$, the number of iterations $iter \leftarrow 0$, the threshold of candidate segment to look for, $dirty\_threshold$

Output: the victim segment number min_segno

1: **while** $iter < dirty\_threshold$ **do**
2: Calculate the age of the candidate segment, $age \leftarrow 10000 \times \frac{max\_mtime - mtime}{total\_time} \times age\_weight$;
3: Calculate the free space that can be obtained after migrating the segment, $u \leftarrow 10000 \times \frac{512 - vblocks}{512} \times (100 - age\_weight - f\_weight)$;
4: Adjust f of the victim candidate segment, $f \leftarrow 10000 \times \frac{f}{max\_f - min\_f} \times f\_weight$;
5: Calculate the migration cost, $cost \leftarrow UINT\_MAX - (age + f + u)$;
6: Increase the number of iterations, $iter++$;
7: **if** $cost < min\_cost$ or ($cost == min\_cost$ and $age > oldest\_age$) **then**
8: Update the minimum migration cost, $min\_cost \leftarrow cost$;
9: Update the max age, $oldest\_age \leftarrow age$;
10: Update the victim segment number with the least migration cost, $min\_segno \leftarrow segno$;
11: **end if**
12: **end while**
13: Select the segment with the same age as the victim segment as the target segment;
14: Migrate valid blocks on the victim segment to the target segment through threaded logging writes;

III. Evaluation of FAGC

Li et al. propose a multi-level threshold synchronous write technology and a high-frequency detection background segment cleaning (MWHFB) to reduce GC overheads [5]. The latest ATGC [3] proposed by the Linux community optimizes F2FS GC. We use them and traditional F2FS as comparison schemes. Traditional F2FS and MWHFB are deployed in Linux kernel 4.9.76, ATGC and FAGC are deployed in Linux kernel 5.4.147.

We imitate the evaluation method of ATGC [3] and write 7,000 dirty segments of three types respectively. The f of segments type A is small while that of type B is large, and the number of valid blocks on segments A and B is the same. Segments of type C are target segments where valid blocks of segments A and B are migrated to. The proportion of invalid data for this synthetic benchmark is 55.53%. The number of GCs, valid blocks migrated, and free space fragments, and the capacity of free space fragments are shown in Table I. FAGC reduces the number of GCs by 82.68% and 74.51% than traditional F2FS and ATGC, respectively. The number of valid blocks migrated by FAGC is reduced by 82.82% and 73.92% correspondingly. FAGC still cleans up free space fragments despite its GC reduction. Each GC in FAGC selects the segment with severe free space fragmentation and reduces the free space fragmentation. We take the minimum energy consumption of BGGC we measured, 102.79 mJ per BGGC. Traditional F2FS, MWHFB, ATGC, and FAGC consume 198.8 J, 216.37 J, 135.07 J, and 34.43 J, respectively. FAGC consumes 164.37 J and 100.64 J less than traditional F2FS and ATGC to complete this synthetic benchmark, respectively. FAGC eliminates the need to recycle some segments that meet BGGC valid blocks and age requirements because recycling these segments cannot reduce free space fragments.

TABLE I: GC and free space fragmentation

Scheme	GC count	Valid blocks	# of free space fragments	Capacity (MB)
F2FS	1,934	450,790	3,187	64.65
MWHFB	2,105	479,288	3,010	62.29
ATGC	1,314	296,956	3,045	62.78
FAGC	335	77,434	3,282	63.71

References

[1] C. Lee, D. Sim, J. Hwang, and S. Cho, "F2fs: A new file system for flash storage," in *13th USENIX Conference on File and Storage Technologies (FAST)*, pp. 273–286, 2015.

[2] L. Yang, F. Wang, Z. Tan, *et al.*, "Ars: Reducing f2fs fragmentation for smartphones using decision trees," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1061–1066, 2020.

[3] C. Yu, "f2fs: support age threshold based garbage collection," 2021.

[4] L. Huawei Technologies Co., "Kirin 9000," 2022.

[5] Q. Li, A. Deng, *et al.*, "Optimizing fragmentation and segment cleaning for cps based storage devices," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC)*, pp. 242–249, 2019.