A New Technology for Real-Time File System of High-Speed Storage System in Airborne sensors

Haishan Tian¹, Fangfang Ju⁹(a), Hongshan Nie²(b), Yuanyu Wu¹, Qiong Yang¹, and Shuangjian Li¹

Abstract The airborne sensors with high data rate demand the onboard high-speed storage system due to the limited bandwidth of the downlinks. The NAND Flash and its products are the most common storage medium, and the conventional FAT file system is widely used in the storage systems. However, there has been a problem in recording the live high-speed data stably with this file system. The management time of the FAT file system is relatively long due to the internal overhead which includes the retrieving and updating of the FAT and FDT. The aim of this study is to investigate the technology for the real-time file system. A technology called FPFPA (FAT post- and FDT post-allocation) method is presented to solve this problem. To evaluate the performance of the proposed method, the management time of the file system using our proposed method is tested on the high-speed storage system of an airborne radar. The result indicates that the proposed method achieves our goal successfully in that the management time of the file system is significantly reduced and sufficiently small.

Keywords: airborne sensor; file system; real-time; high-speed storage.
Classification: Circuits and modules for storage.

1. Introduction

The data rate of the airborne sensors is usually hundreds of Mb/s or several Gb/s [1,2,3]. This demands onboard data storage system with the high speed, because the data transfer rate of the aerial vehicles’ downlink can hardly achieve over 400 Mb/s [4,5,6,7].

With the advantages of shock resistance, large capacity, and fast data reading/writing rate, the products based on NAND Flash, such as solid-state disk and compact flash (CF), are usually used as the medium of the storage system [8,9,10]. Meanwhile, the file system is required to organize the storage effectively. The conventional file allocation table (FAT) file system has been widely used in the airborne storage systems. However, it is not originally designed for the flash products, but for magnetic storages. The file management time of FAT system, which includes the retrieving and updating time for controlling the data files, is relatively long, which becomes an obstacle to the high-speed storage.

To reduce the management time of the FAT file system in the embedded storage systems, many revised methods have been proposed by different scholars [11]. For example, a method called new FAT file system (NFAT) is presented by Monsoon Choi. The NFAT could reduce the data retrieving time for searching free cluster in FAT [12]. However, the updating time of FAT and file description table (FDT) is still long in NFAT, which significantly decreases the storage speed. To reduce the updating operation of the file system, the all clusters pre-allocation (ACPA) method is proposed and pre-allocates all the available clusters when a file is created [13]. Although ACPA method avoids frequent FAT updating, the file management time is still long when frequently pre-allocating and releasing many free clusters in the storage system. The FAT pre- and FDT quasi-allocation (FPFQA) method greatly compresses file management time [14,15], but all storage space is pre-allocated in this method, which results in the inability to modify any file information during the storage process. Therefore, this method is only suitable for the mass data storage, not for normal applications.

The technology called FPFPA (FAT post- and FDT post-allocation) is proposed to remove or reduce the retrieving and updating of the FAT and FDT in this study. To evaluate the performance of our new technology, we modified the FAT file system with the FPFPA method, and implemented the prototype real-time FAT file system on the miniature high-speed data storage system of an airborne frequency modulated continuous wave synthetic aperture radar (FMCW SAR).

The remaining sections of the study are organized as follows. The background of the FAT file system is described in Section 2. The FPFPA method is presented to solve the problem in Section 3. Section 4 evaluates the performance of the proposed method. Finally, conclusions are drawn in Section 5.

II. Background

The three main components of the FAT file system are FAT, FDT, and data region [16,17,18,19], as shown in Fig. 1.
The FAT is the core of the file system. It is used to manage and describe the internal space of the files in the data storage. The space with the size of 4B in FAT is used to describe the linking relationship of a cluster. Two duplicated FATs are maintained for robustness in the FAT file system [20,21,22,23].

The FDT contains the directory information of the files in the storage medium. It uses space with the size of 32B to describe the information of one file, such as the file name, size, type, storage starting position and time [24,25,26].

The data region is the area where the data are actually stored and occupies the main space of the storage medium. It is composed of a large number of clusters which contains dozens of sectors [27,28,29,30]. The sector is the basic unit of data reading and writing, and its size is 512 B.

In the FAT file system, with the data of one cluster stored in the data region, there are four file management operations in addition to the actual data writing: the retrieving for a free cluster in FAT, the updating of a cluster link relationship in FAT, the retrieving for the file information in FDT and the updating of file information in FDT. Therefore, the total time is

$$T_{write\_cluster} = T_{data\_write} + T_{FAT\_read} + T_{FAT\_write} + T_{FDT\_read} + T_{FDT\_write}$$

(1)

where \(T_{data\_write}\) is the data writing time of one cluster, \(T_{FAT\_read}\) and \(T_{FDT\_read}\) is the retrieving time of the FAT and FDT, respectively. \(T_{FAT\_write}\) and \(T_{FDT\_write}\) is the updating time of the FAT and FDT, respectively.

Because the cluster link information is searched sequentially in the FAT file system, the size of the data for retrieving FAT ranges from one sector to the total sectors of the FAT. So \(T_{FAT\_read}\) is

$$T_{FAT\_read} = n \times [(512 \times T_r) + T_{w\_req}]$$

(2)

where \(n\) is the number of the retrieving sectors, \(T_r\) are the average reading time of 1 B data, \(T_{w\_req}\) is the average response time of the reading request.

Although only a cluster link information with the size of 4 B is required to be written into FAT, one sector’s data need to be updated each FAT, because the basic unit for the data writing operation is the sector. So \(T_{FAT\_write}\) is

$$T_{FAT\_write} = 2 \times [(512 \times T_w) + T_{w\_req}]$$

(3)

where \(T_w\) are the average writing time of 1 B data, which is slightly larger than \(T_r\). \(T_{w\_req}\) is the average response time of the writing request.

As we know, NAND flash memory has some limitations, such as erase-before-write and limited erase count, the \(T_r\) and \(T_w\) are the time which have already taken these characteristics of the NAND flash in account.

The file is also searched sequentially in FDT, so the \(T_{FDT\_read}\) is

$$T_{FDT\_read} = m \times [(512 \times T_r) + T_{w\_req}]$$

(4)

where \(m\) is the number of the retrieving sectors in FDT.

Although only a directory entry with a size of 32 B needs to be updated, one sector’s data are also needed to be written to the FDT. So \(T_{FDT\_write}\) is:

$$T_{FDT\_write} = 512 \times T_w + T_{w\_req}$$

(5)

The ratio of the file management time to the data writing time is called \(Rate_{ed}\) in this study. The value of \(Rate_{ed}\) is equal to:

$$Rate_{ed} = \frac{T_{FAT\_read} + T_{FAT\_write} + T_{FDT\_read} + T_{FDT\_write}}{T_{data\_write}}$$

$$= \frac{3 \times [(512 \times T_w) + T_{w\_req}] + (n + m) \times [(512 \times T_r) + T_{w\_req}]}{k \times T_w + T_{w\_req}}$$

(6)

where \(k\) is the size of one cluster.

It can be seen that the file management time cannot be ignored in comparison with the real data writing time. Sometimes the value of \(Rate_{ed}\) can even exceed dozens. Therefore, the file management of the original FAT could significantly decrease the speed of the data storage.

### III. FPFPA Method

The FAT file system is not suitable for directly applying to the high-speed data storage. Although there is a FTL (flash translation layer) which support the FAT file system, the FTL cannot be modified in many NAND flash products. So, a FPFPA method is proposed to reduce the overhead of file system in this study. Four proposals are presented in this method.

First, all the clusters are linked sequentially in FAT, and this proposal is called FAT sequentially link method. The used cluster numbers are got by retrieving the FAT before the storage starts. During the data storage, the data are sequentially written into the free clusters of the data region and their cluster numbers can be obtained by calculating the amount of the stored data. Therefore, the operation for retrieving free clusters is not needed in the data storage, i.e., \(T_{FAT\_read}\) is eliminated.

Fig. 2 shows how the retrieving of the FAT is eliminated.
using this method. Fig. 2 (a) is the FAT of a blank storage medium, where “F8FF FF0F FFFF FFFF” identifies the starting position of the FAT and Cluster_N is the total number of clusters. According to this method, the data storage could start from any cluster number N indicated by the arrow. The data are recorded in sequence until the end position of storage medium. Then the storage position returns to the cluster D + 1, where cluster 2 to cluster D are the space allocated for the FDT. The data are continually recorded in sequence until there is no storage space. The purpose of starting storage from any cluster N is to make the probability of storage device space being used equally, and to ensure the service life of the storage medium. Fig. 2 (b) denotes the partly used FAT, where cluster N + M is the last used cluster and “FF0F FFFF” identifies the end position of a file. The difference between the operations shown in Fig. 2 (a) and (b) is only that their starting storage positions differ. The partly used storage medium starts recording data from cluster N+M+1. As shown in two figures, the retrieving of the FAT is only operated before the data storage, and the cluster number can be calculated according to the amount of the stored data, without need to query the FAT during data storage.

**Fig.2.** FAT sequentially link method

Second, the clusters of the file are post allocated in FAT, and this proposal is called FAT post-allocation method. The cluster link information is only updated after the data of one file are completely written into the data region, which greatly reduces \( T_{FAT\_write} \). Fig. 3 shows how the FAT is post allocated using this proposal, where \( N+U \) is the last used cluster number. Only when all the data of the file \( K \) are stored in the data region, the cluster updating starts from the first unused cluster \( N+U+I \), as indicated by the arrow in the Fig. 3, and ends at the cluster \( N+U+F \) of the file. Where \( F \) is the number of clusters contained in the file \( K \). This proposal allows the FAT to be updated only once after one file is stored, thus reducing the updating time of the FAT to \( 1/F \) of that in the original FAT file system.

**Fig.3.** FAT Post-Allocation method

Third, the file information is sequentially stored in FDT, and it is called FDT sequentially store method. Specifically, the last used directory entry of the file is obtained by retrieving the FDT before the data storage, the subsequent clusters are all free and then store the file information in order during the storage. Therefore, there is no need to query the FDT, i.e., \( T_{FDT\_read} \) is removed. Fig. 4 shows how the retrieving of the FDT is eliminated using this proposal. Fig. 4 (a) is the FDT when the storage medium is blank. There is not any file’s information inside. During the data storage, the file information is written sequentially from the beginning of the FDT. Fig. 4 (b) depicts partly used FDT, where \( S \) represents the number of the used files. During the data storage, the file information is written sequentially from the \((S+1)\)th file, as indicated by the arrow in the Fig. 4. Therefore, the querying of the FDT is not needed.

**Fig.4.** FDT sequentially store method: (a) Blank FDT; (b) Partly used FDT

Finally, the information of 16 files is post allocated in the FDT whenever the data of the 16 files are stored, and this proposal is called FDT post-allocation method. Therefore, \( T_{FDT\_write} \) is reduced to \( 1/(16\text{files’ cluster numbers}) \) of the original FDT updating time. Fig. 5 shows how the FDT is post allocated in this proposal. Fig. 5(a) denotes the FDT after each 16 files’ data are written into the data storage. The 16 files’ information is updated simultaneously in FDT because it is described by the data of 1 sector, and the sector is the smallest unit of reading and writing. If the number of the data files is less than 16, the last \( M \) files’ information is updated, as shown in Fig. 5(b).
IV. Experimental Result

The real-time file system with the FPFPA method is implemented on a miniature high-speed mass storage system of an airborne sensor, as shown in Fig. 7. The red rectangle in the right of the figure is the storage system which is mounted on an airborne radar. The FPGA is the host of the CF card which is one of the NAND Flash productions. The maximum data writing rate of the CF card is 160MB/s, and its storage is 128 GB. To evaluate the performance of the FPFPA method, we measured the file management time of the following four file systems: (1) the original FAT file system, (2) the file system with the ACPA method, (3) the file system with FPFQA method, (4) the system with our proposed FPFPA method.

![Fig.7. The miniature high-speed mass storage system of an airborne radar](image)

Because the $Rate_{td}$, as shown in formula (6), represents the proportion of the file management time to the data writing time, it is used to evaluate the performance of different file systems. The test methods for measuring and calculating $Rate_{td}$ of four file systems vary due to the difference of their working mechanism.

To measure the $Rate_{td}$ of the original FAT file system, the size of each writing request is a cluster with the size of 128 KB, and 16 K sequential writing requests were generated. Fig. 8 denotes the test result. The vertical axis represents $Rate_{td}$ of each writing requests. The range of the result is from 36 to 151 and the average value is about 93, so the $Rate_{td}$ of the original FAT file system is large. The value of the result increases linearly, because the size of the retrieving in FAT and FDT grow with the amount of the data storage.

![Fig.8. The $Rate_{td}$ of the FAT file system](image)
The \( \text{Rate}_{f,d} \) of the ACPA method is proportional to the size of free space. Taking the file with the size of 64 MB as an example, Fig. 9 depicts the measured and calculated \( \text{Rate}_{f,d} \) with the ACPA method, where the vertical axis represents \( \text{Rate}_{f,d} \) of a file. The value of result is from \( 3.1 \times 10^{-3} \) to 0.63 and the average value is about 0.315.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{fig9.png}
\caption{The \( \text{Rate}_{f,d} \) of the file system with the ACPA method.}
\end{figure}

To measure \( \text{Rate}_{f,d} \) of the file systems with the FPFQA method and the FPFPA method, taking the file with the size of 256 MB as an example, the data of 128 GB are written to the CF card. Fig. 10 denotes the result using FPFQA method, the horizontal axis is the size of the data storage, and the vertical axis represents \( \text{Rate}_{f,d} \). It is shown that \( \text{Rate}_{f,d} \) is sufficiently small.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{fig10.png}
\caption{The \( \text{Rate}_{f,d} \) of the file system with the FPFQA method.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{fig11.png}
\caption{The \( \text{Rate}_{f,d} \) of the file system with FPFPA method.}
\end{figure}

Table 1 shows the average value of the \( \text{Rate}_{f,d} \) using the four file management methods.

| Method          | Original FAT system | ACPA method | FPFQA method | FPFPA method |
|-----------------|---------------------|-------------|--------------|--------------|
| Average value of \( \text{Rate}_{f,d} \) | 93 \( \times 10^{-3} \) | 0.315 | < \( 5 \times 10^{-4} \) | < 0.027 |

Compared with other file systems, the \( \text{Rate}_{f,d} \) of the FPFPA method is much smaller than that of the FAT file system and the ACPA method. Although it is larger than that of the FPFQA method, the storage speed decreases only 2.63% at most. And the space of the storage medium is not required to be pre allocated. Therefore, the file management time of the FPFPA method is sufficiently small, and the FPFPA can be commonly used in the high-speed airborne storage.

V. Conclusion

This paper investigates the technology for real-time file system in airborne high-speed embedded storage. The problems for the file management are analyzed in the airborne application which has the many special requirements such as the fast reading/writing speed and shock resistance. The retrieving and updating time of the FAT and the FDT is relatively long. The called FPFPA is proposed to reduce the file management time. The performance of the FPFPA method is evaluated in the storage system of an airborne radar. The results indicate that the capability of the real-time file management is achieved. The FPFPA method can be widely used in the airborne storage system.

Acknowledgments

This research was funded by Doctoral Science Foundation of Hunan Normal University grant number 0531120 and
General Project of Hunan Provincial Department of Education grant number 19C1165.

References

[1] H. Bi, et al.: “FM-CW SAR Sparse Imaging Based on Approximated Observation: An Overview on Current Technologies,” IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 13(2020)4825 (DOI: 10.1109/JSTARS.2020.3017487)

[2] M.L. Ding, et al.: “A W-Band 3-D Integrated Mini-SAR System With High Imaging Resolution on UAV Platform,” IEEE Access, 8(2020)113601 (DOI: 10.1109/ACCESS.2020.3002723).

[3] P. Stephan, et al.: “Airborne Circular W-Band SAR for Multiple Aspect Urban Site Monitoring,” IEEE Transactions on Geoscience and Remote Sensing, 4(2019)9696 (DOI: 10.1109/TGRS.2019.2909949).

[4] H. Horst, et al.: “Simulation-Based Signature Analysis of Fuel Storage Tanks in High-Resolution SAR Images,” IEEE Geoscience and Remote Sensing Letters, 14(2017)1278 (DOI: 10.1109/LGRS.2017.2706421).

[5] H.S. Tian, et al.: “A Compact High-Speed Storage System for an Airborne SAR,” in Proc. International Conference on Modern Circuits and Systems Technologies(2016) (DOI: 10.1109/MOCAST.2016.7495114).

[6] N. Gianluca, et al.: “Impact of Cycling on Random Telegraph Noise in 3-D NAND Flash Arrays,” IEEE Electron Device Letters, 39(2018)1175 (DOI: 10.1109/LED.2018.2847341).

[7] M.E. Yanik, et al.: “Development and Demonstration of MIMO-SAR mmWave Imaging Testbeds,” IEEE Access, 8(2020)126019 (DOI: 10.1109/ACCESS.2020.3007877).

[8] D. Yimo, et al.: “WeLe-RAID: a SSD-based RAID for system endurance and performance,” in Proc. of the IFIP International Conferencee Network and Parallel Computing(NPC 2011), (2011)755-758.

[9] Y.M. Du, et al.: “CSWL: Cross-ssd wear-leveling method in ssd-based raid systems for system endurance and performance,” Journal of Computer Science and Technology, 28(2013)328.

[10] J.Q. Zhang, et al.: “WB-RAIS: White-Box Redundant Array of Independent SSDs,” Journal of Wireless Personal Communications, 10(2018)25.

[11] C. F. Gu, et al.: “Multi-core DSP based parallel architecture for FM-CW SAR real-time imaging,” Radioengineering, 24(2015)1084.

[12] M. Choi, et al.: “Design and implementation of a FAT file system for reduced cluster switching overhead,” in Proc. Int. Conf. on Multimedia and Ubiquitous Engineering, (2008)886-889 (DOI: 10.1109/MUe.2008.42).

[13] H. Park, et al.: “New techniques for real-time FAT file system in mobile multimedia devices,” IEEE Transaction on Consumer and Electronic, 52,(2006)9 (DOI: 10.1109/TCE.2006.1605017).

[14] H.S. Tian, et al.: “Study on an on-board data storage system for FM-CW SAR,” Journal of Applied Remote Sensing, 10(2016)11.

[15] H.S. Tian, et al.: “Integrated System of Mini-SAR Real-Time Signal Processing and Data Storage,” IEEE Radar Conference, (2015)66-69 (DOI: 10.1109/RadarConf.2015.7411907).

[16] A. Boubriak, et al.: “SlimFS: A Thin and Unobtrusive File System for Embedded Systems and Consumer Products,” IEEE Transactions on Consumer Electronics, 64(2018) 334 (DOI: 10.1109/TCE.2018.2867826).

[17] G.T. Raul, et al.: “BenchBox: A User-Driven Benchmarking Framework for Fat-Client Storage Systems,” IEEE Transactions on Parallel and Distributed Systems, 29(2018)2191 (DOI: 10.1109/TPDS.2018.2819657).

[18] S.H. Chen, et al.: “A Partial Page Cache Strategy for NVRAM-Based Storage Devices,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 39(2020)373 (DOI: 10.1109/TCAD.2018.2887045).

[19] I. Shin: “Improving internal parallelism of solid state drives with selective multi-plane operation,” Electron. Lett. 54 (2018) 64 (DOI: 10.1049/el.2017. 3912).

[20] G.W. Jia, et al.: “Simplified Real-Time Imaging Flow for High-Resolution FM-CW SAR,” IEEE Geoscience and Remote Sensing Letters, 12(2015)973-977 (DOI: 10.1109/LGRS.2014.2370733).

[21] G. Nicosia, et al.: “Impact of Cycling on Random Telegraph Noise in 3-D NAND Flash Arrays,” IEEE Electron Device Letters, 39(2018)1175 (DOI: 10.1109/LED.2018.2847341).

[22] J. Zhang, et al.: “WB-RAIS: White-Box Redundant Array of Independent SSDs,” Journal of Wireless Personal Communications, 10(2018)15 (DOI: 10.1109/LED.2018.2957341).

[23] J. Li, et al.: “Patch-Based Data Management for Dual-Copy Buffers in RAID-Enabled SSDs,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 39(2020)3956 (DOI: 10.1109/TCAD.2020.3012252).

[24] Q. Li, et al.: “Joint Decoder Design for SSDs,” IEEE Transactions on Signal Processing, 69(2019)2399 (DOI: 10.1109/TSP.2019.2905822).

[25] Y. Son, et al.: “Design and Implementation of SSD-Assisted Backup and Recovery for Database Systems,” IEEE Transactions on Knowledge and Data Engineering, 32(2020)260 (DOI: 10.1109/TKDE.2018.2884466).

[26] C. Zambelli, et al.: “Modeling the Endurance Reliability of Intradisk RAID Solutions for Mid-IX TLC NAND Flash Solid-State Drives,” IEEE Transactions on Device and Materials Reliability, 17(2017)13 (DOI: 10.1109/TDMR.2017.2749639).

[27] S.R. Mashwani, et al.: “360° Semantic File System: Augmented Directory Navigation for Nonhierarchical Retrieval of Files,” IEEE Access, 7(2018)9406 (DOI: 10.1109/ACCESS.2018.2890165).

[28] N.R. Paiker, et al.: “Design and Implementation of an Overlay File System for Cloud-Assisted Mobile Apps,” IEEE Transactions on Cloud Computing, 8(2020)97 (DOI: 10.1109/TCC.2017.2763158).

[29] S.B He, et al.: “HARL: Optimizing Parallel File Systems with Heterogeneity-Aware Region-Level Data Layout,” IEEE Transactions on Computers, 66(2017)1048 (DOI: 10.1109/TC.2016.2637905).

[30] S.Z.Wu, et al.: “PPF: Improving the Reliability of Deduplication-based Storage Systems with Per-File Parity,” IEEE Transactions on Parallel and Distributed Systems, 30(2019)2117 (DOI: 10.1109/TPDS.2019.2898942).