

Improving File System Performance and Reliability of Car Digital Video Recorders

Younghun Kim, Dongkun Shin, Member, IEEE

Abstract — Car digital video recorders (DVRs) store real-time audio/video data at flash memory storage device. Car DVR should handle a large amount of high quality multimedia data with high reliability. However, current file systems are vulnerable to sudden-power-offs and flash memory errors. This paper proposes flash-aware cluster allocation techniques and a flash-aware journaling technique. The flash-aware cluster allocation techniques can reduce the probability of file system corruptions and file fragmentations. The flash-aware journaling technique can ensure the file system consistency without significant overheads. Experiments on a simulator and a real system showed that the proposed techniques can prevent file fragmentations and file system corruptions, and they can improve the write performance and lifetime of SD card significantly¹.

Index Terms — FAT file system, Car digital video recorder, Cluster allocation, Flash-aware journaling.

I. INTRODUCTION

Car digital video recorder (DVR), which is also called car black box or dashboard camera, records real-time audio/video data at every moment of drive. It can also record audio/video data at every accidental or suspicious event of a parked car. Generally, the captured multimedia data are saved in a secure digital (SD) card, which uses NAND flash memory as storage media. The SD card is formatted with the PC-compatible FAT file system. The car DVR should provide a high reliability on recording the captured data since it can be used for evidence in legal action and the car DVR is exposed to system failures such as sudden-power-off (SPO). However, the current file systems used in car DVRs are not designed considering the required high reliability.

First, the current FAT file system of car DVR system updates metadata frequently. The FAT file system metadata includes the file allocation table (FAT) and the directory entry (DE) [1]. The FAT file system manages the storage space in the unit of cluster, which is the basic unit for space allocation. The FAT consists of an array of entries, each of which represents the allocation status of the corresponding cluster. The FAT entry also has the cluster chain information for

created files. The FAT is modified when any cluster is allocated or freed. The DE has the file information such as file name, file size, first cluster number, date/time, etc. The DEs are modified when files are created, appended, or deleted. Since the car DVR makes a new file at every predefined time intervals and ceaselessly records the driving data, the metadata are frequently updated. If the metadata are corrupted by SPO or flash memory errors, the recorded data cannot be accessed. The frequent metadata updates increase the probability of file system corruptions.

In particular, there is a mismatch between the size of FAT entry and the size of flash memory page. Whereas an FAT entry is 4 byte, the flash memory page is generally 4 KB or 8 KB. Since the flash memory can be programmed by the unit of page, the whole flash page should be reprogrammed even though the file system wants to modify only a 4-byte FAT entry [2]. In addition, the minimum data transfer unit between file system and storage device is essentially a 512-byte of sector. Since flash memory does not permit in-place update, a read-modify-write (RMW) operation is required to update only a partial page, that is, the old flash page is first read into the host buffer, several sectors are modified in the buffer, and the modified page data are written at a new flash page. The embedded firmware of SD card, called flash translation layer (FTL), performs the RMW operations. The RMW will increase the latency of FAT update operation. In addition, SD card has a limited program/erase (P/E) cycles. The small write requests requiring the RMW operations will exhaust rapidly the P/E cycles of SD card. Some flash memory devices permit the partial page programming, and thus they can reduce the number of RMW operations for updating FAT. However, when there is an SPO or flash page program error, the partial page programming can corrupt other FAT entries that are previously written at the target flash page [3].

The second problem is the file fragmentation. Car DVR generates several different types of data and places them in different folders. The data can be categorized based on when they are generated. For example, the continuously recorded video/audio files during driving are placed into the “driving” folder. The vehicle accidents and driving events triggered by the internal gyroscope are placed into the “driving event” folder. The videos triggered by the motion detection and collision detection during parking are stored into the “parking motion” and “parking collision” folders, respectively. Each directory has its limited space and is recycled automatically when the directory space becomes full. The oldest files are overwritten first. Therefore, the files in different directories

¹This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2013R1A1A2A10013598).

Younghun Kim and Dongkun Shin are with the College of Information and Communication Engineering, Sungkyunkwan University, Suwon, Korea (e-mail: ehdeoddl, dongkun@skku.edu).

have different lifetimes and file sizes. Whereas these directories are separated logically, the data of a file can be written at any cluster in the file system irrespective of the file type. Generally, the FAT file system allocates the first free cluster. Since the lifetimes of recorded files are different, free clusters are fragmented and a newly created file will be written into several fragmented clusters.

The file fragmentation incurs random write operations at SD card. Since flash memory storage show worse performance for random writes compared with sequential writes, the file fragmentation will degrade the file system write performance, and can invoke frame drops at the worst case [4], [5]. In addition, since the random write requests invoke significant write amplification within flash memory device, the lifetime of SD card will be shortened by the fragmented write requests. Unfortunately, any defragmentation technique cannot be a solution. Since the car DVRs should ceaselessly write the captured data at SD card, it has no idle time for defragmentation job.

Kim and Shin [6] proposed a partitioning technique which can solve the fragmentation problem. However, the technique cannot guarantee the file system consistency at system failures. In this paper, the partitioning technique is extended and a new journaling technique is proposed for high performance and high reliable car DVR systems. First, the partitioned cluster allocation (PCA) technique divides the storage space into multiple regions and allocates the clusters of same-type of files from the same region in order to prevent file fragmentation. Second, the page-aligned pre-allocation (PAPA) technique pre-allocates the clusters of created files to prevent flash memory pages allocated for FAT from being overwritten multiple times. By the pre-allocation, the number of metadata updates is also reduced, and thus the file system is unlikely to be exposed to the failure critical situations. Third, the flash-aware journaling (FAJ) ensures the file system consistency by logging the metadata changes at the journal area. The FAJ is designed to consider the characteristic of flash memory, and it has a minimal overhead for managing the journal. Experiments are performed at a car DVR simulator and a real car DVR system. By the proposed techniques, the file fragmentations are almost removed and the metadata updates are significantly reduced.

The remainder of the paper is organized as follows. Section 2 introduces related works. Section 3 explains the proposed techniques in details. Section 4 shows the experimental results. Finally, Section 5 concludes the paper

II. RELATED WORKS

Several optimization techniques have been proposed to increase the performance of FAT file systems. The all cluster pre-allocation (ACPA) [7] technique allocates all the clusters for a created file at once. Therefore, it can remove the updates on FAT during the append operations of the file. Since the exact file size is unknown at the file creation, ACPA pre-allocates all the free clusters in the system for the created file, and de-allocates unused clusters at file close. Therefore, many FAT entries are allocated and de-allocated repeatedly, and many write operations will be sent to the flash memory to

update the FAT. Moreover, ACPA does not consider the partial page programming issue and cannot handle the file fragmentation problem.

The flash-aware extension-based cluster allocation (FECA) [8] is an anti-fragmentation technique. FECA divides the storage space into multiple cluster groups, where the size of a cluster group is same to the size of flash memory block. FECA allocates free clusters from different cluster groups depending on the file size. Therefore, large files and small files are not mixed within a same cluster group. Since different sizes of files generally have different lifetimes, FECA can mitigate the file system fragmentations. However, when the file system utilization is high, the fragmentations are inevitable. A large file can be fragmented into several separated cluster groups, and it will be mixed with small files.

The specification of FAT file system does not include the journaling scheme for improving file system consistency. Therefore, a system crash can corrupt the file system consistency. Kim and Yu [9] proposed a journaling technique for FAT file system, called HFAT. It allocates the journal areas dynamically in order to distribute the frequent write accesses on journal. The total size of journal area also can be resized dynamically. However, the distributed journal areas will aggravate the file fragmentation problem. Moreover, HFAT requires a large storage space overhead for journal area.

Alei *et al.* [10] proposed the metadata delayed sequential write technique, which collects a certain amount of metadata updates and records them at one time. The delayed sequential write technique can reduce the number of metadata updates without breaking the file system consistency. However, under the delayed write technique, the latest unwritten data will be lost at SPO. In addition, the technique cannot guarantee the file system consistency at sudden system crashes.

There are also FTL-level optimization techniques for improving the performance of flash memory storage devices. Ryu [11] proposed the filtering FTL technique (FFTL), which filters metadata updates and manages them separately from normal data writes. FFTL reduces the block erase count and improves the write performance of metadata update. However, it can reduce neither the number of metadata updates nor the probability of file system corruptions. Kim *et al.* [12] proposed the page padding technique in order to provide a constant write performance even for the fragmented case. It changes a fragmented write pattern to a sequential write pattern by padding existing data in FTL level. However, the page padding technique can be useful only when the fragmented clusters are located closely.

Therefore, the previous techniques can handle only a subset of the FAT file system problems, and they did not consider the workloads of car DVR systems.

III. OPTIMIZATION TECHNIQUES

A. Partitioned Cluster Allocation

When free clusters in FAT file system are fragmented, a newly created file will be allocated with the fragmented clusters. The fragmented free clusters are generated due to the different lifetimes and sizes of car DVR files. When two different types of files are allocated with adjacent clusters, the

fragmented free clusters are generated if only one of them is deleted and its clusters are de-allocated.

The file fragmentation problem may be solved by a defragmentation technique, which moves the data in fragmented clusters into the contiguous clusters. However, the defragmentation technique cannot be applied to the car DVR systems since it invokes a runtime overhead. In order to move the data at fragmented clusters, additional read and write operations should be sent to SD card. Since the car DVRs should ceaselessly write the captured data at SD card, the additional operations will result in frame drops due to the limited I/O bandwidth of the storage. In addition, the lifetime of SD card will be shortened. Therefore, the car DVR systems need an anti-fragmentation technique rather than defragmentation.

The proposed *partitioned cluster allocation* (PCA) technique is an anti-fragmentation technique, which can prevent file fragmentations by dividing the storage space into several regions and storing only a same type of files at each region. The approach of PCA is different with the legacy logical or physical storage partitioning techniques. The logical partition, which is managed by file systems, cannot solve the file fragmentation problem since different partitions share the physical clusters. On the other hand, the physical partition can separate the clusters of different partitions. However, each partition should be mounted with a separated file system. In addition, each partition size cannot be changed dynamically. Moreover, some operating systems do not permit multiple partitions on one removable storage device.

Although the proposed partitioned cluster allocation uses a special cluster allocation technique in order to remove the file fragmentations, it still provides the compatibility with the current FAT file system. Fig. 1 shows the cluster allocations under the PCA technique. The storage space is divided into several regions, each of which is specified with the start cluster number, the end cluster number, the last allocated cluster number, the region size, and the target file type. The start and end cluster numbers determine the size and location of a region. Based on the predefined space portion of each region and the total storage size, the start and end cluster numbers of each region are calculated by file system during the file system mount process. The information of each region is saved at a special configuration file that is created by the car DVR application. The file system maintains the region information in main memory during runtime.

Each region can be used for only single type of files. Therefore, different types of files will not be mixed. The file type can be identified from the directory information of the file. When the cluster allocation requests for a file is sent to the file system, PCA allocates free clusters from the region of the file type. The free clusters can be allocated from its region in a circular manner. Since the car DVR system generates a new multimedia file at every time intervals, only recently created files are stored at SD card, and the old files in the target directory are deleted when the directory space becomes full. Therefore, each region can be managed in the round-

robin fashion. Therefore, the next cluster of the last allocated cluster will always be a free cluster. Starting from the location, the PCA allocates contiguous clusters for a new file without fragmentations. Therefore, PCA can remove the free cluster searching overhead. In Fig. 1, for example, when a file with type 1 is created, contiguous free clusters can be found within a constant time without scanning the target region from the cluster number 1 to the cluster number 12.

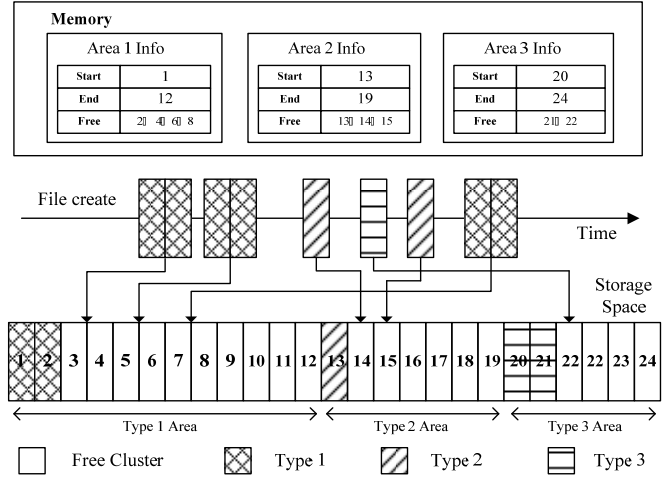


Fig. 1. Cluster allocations in the partitioned cluster allocation technique.

PCA does not change the storage format of FAT file system. Therefore, the files generated under the PCA technique can be accessed by other systems such as host PC. PCA technique can increase the lifetime of SD card as well as the write performance by reducing the number of write requests on the storage.

B. Page-aligned Pre-allocation

Generally, car DVR system buffers captured data in main memory and it flushes them into the storage periodically. As shown in Fig. 2(a), each buffer flush operation updates the metadata such as FAT and DE of the target file since new clusters are allocated and the file size is modified. Therefore, the FAT and DEs are frequently updated.

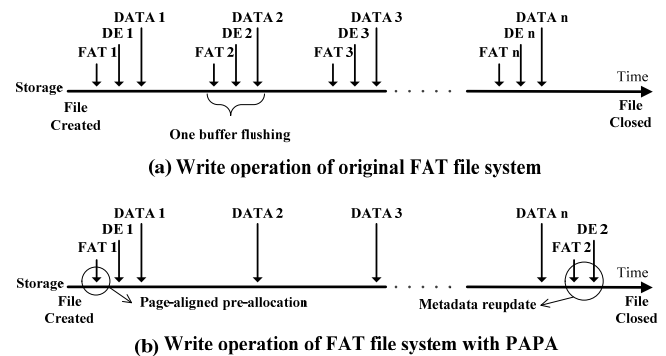


Fig. 2. Write operations while recording a captured data.

The proposed *page-aligned pre-allocation* (PAPA) technique requires only one or two times of metadata updates regardless of the number of data flushes, as shown in Fig. 2(b). In car DVR systems, the maximum file size of each type of files can be known based on the system configurations such as

the recording time and the file format. Therefore, when a file is created, PAPA pre-allocates the clusters of the file based on the maximum file size, and the corresponding FAT entries are modified. The directory entry of the target file is also fixed at the file creation time since the file size is determined. Then, no metadata is updated during the file append operations. Since the file sizes of a same type of files are not different significantly, the difference between the maximum file size and the real file size is not too large. PAPA uses a special file to store the maximum file sizes of different types of files. When the file is closed, the metadata can be updated with the number of used clusters among the pre-allocated clusters. In order to reduce the number of metadata updates, the metadata are modified only when the number of unused clusters is larger than a predefined threshold value.

Fig. 3 shows the detailed cluster allocation process of PAPA. Each entry in the FAT is 4 bytes at the FAT32 file system, and it represents the allocation status of the corresponding cluster and the cluster chain. For example, if n -th FAT entry has the value of m in the FAT, the cluster with cluster number n is allocated for a file and the cluster with cluster number m is the next allocated cluster. For the last entry of a file, a special predefined value such as 0xFFFFFFFF is written to denote the end-of-file (EOF). The FAT area is allocated at the address space separated from the normal data clusters. The FAT will be stored at multiple contiguous logical pages. In the example of Fig. 3, the FAT is stored at four logical pages with the logical page numbers (LPNs) from 0 to 3.

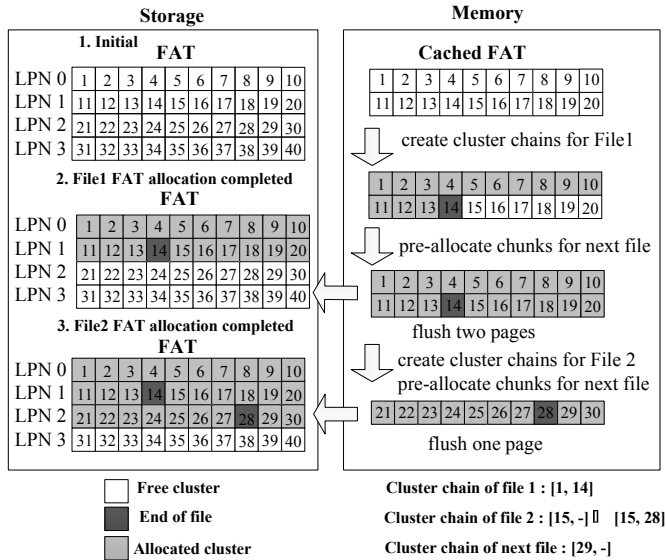


Fig. 3. Cluster allocation process of PAPA.

The PAPA modifies the FAT entries in the unit of flash memory page. Assume that the number of clusters to be pre-allocated for 'File 1' is 14 and one flash memory page can store ten FAT entries. For the file creation request, two flash memory pages with LPN 0 and LPN 1 should be modified by the pre-allocation technique. After closing 'File 1', if 'File 2' is created, the clusters starting from the cluster number 15 will be allocated for the file under the PCA technique, and the flash page with LPN 1 should be modified again. That is, if

the number of FAT entries to be modified by the pre-allocation is unaligned to the flash page size, the logical page that stores the FAT entry for the start cluster should be modified again at each file creation. Such a modification of logical page at file creation will invoke the read-modify-write (RMW) operations or the error-prone sub-page programming within the flash storage.

To resolve the problem, the proposed PAPA technique pre-allocates the clusters such that the FAT is modified in the unit of flash memory page. When PAPA creates the cluster chain for 'File 1', it fills the target page with the cluster chain for the next file in advance, and two pages are written at the storage. When 'File 2' is created, it can use the pre-allocated clusters without modifying the flash page with LPN 1. If 'File 2' requires more clusters, PAPA expands the cluster chain of 'File 2' by writing the next page, which can also have the FAT entries for pre-allocated clusters of another next file. Compared with the previous pre-allocation techniques that pre-allocate the clusters of only the created file, the pre-allocation scheme of PAPA performs the pre-allocation beyond file boundary. Therefore, it can reduce not only the number of FAT updates but also the number of flash page overwrites. Generally, the FTLs of low-end flash memory devices such as SD card use a block-level address translation scheme such as BAST [13] and FAST [14] in order to reduce the size of mapping table. In the block-level translation schemes, the update operations on a same logical page invoke a high garbage collection overhead. Therefore, the proposed PAPA technique can improve the write performance of SD card.

Since PAPA pre-allocates the clusters of not-yet-created files, it should know the pre-allocation information when it allocates the clusters for a new file. The start cluster number and the number of clusters of the next file are maintained in the in-memory file system structure. However, the information will be lost if the system is rebooted without saving it at the storage. Then, PAPA should rebuild the information by scanning the FAT entries. Fig. 4 shows how PAPA finds the clusters pre-allocated for the next file. For each page allocated for FAT, the first FAT entry is examined whether it is allocated or free. If the first entry is allocated, the next page is examined. When a free entry is found from the page with LPN n , the page with LPN $n-1$ is the last modified page. Then, the page is scanned from the last entry to the backward direction until an EOF entry is found. In Fig. 4, the page with LPN 4 is the last updated page, and the FAT entry with cluster number 36 has the EOF. Therefore, the clusters with cluster numbers from 37 to 40 are the pre-allocated clusters of the next file. If a new file requires more number of clusters, PAPA pre-allocates the following clusters modifying the next page allocated for the FAT.

After the pre-allocation, PAPA updates the DE of the created file. The field of file size in DE should be modified according to the number of pre-allocated clusters in order to provide the file system consistency. PAPA ignores all the DE update requests by append operations until the target file is closed. The ignored DE update requests include the

information such as the last access time and file size. The last access time of file can be ignored since the file creation time is sufficient for user. The file size is pre-updated at the file creation time to allow full accesses for the whole file. Therefore, user can access the captured multimedia data without any problem even at SPOs.

If there are many unused pre-allocated clusters when a file is closed, PAPA de-allocates the unused clusters and changes the file size. Generally, if there is an event during driving, the car DVR closes the currently recording normal file in the “driving” folder, and creates a new file at the “driving event” folder. Then, the normal file will leave many unused clusters. In such a case, PAPA allocates the unused clusters for the next file.

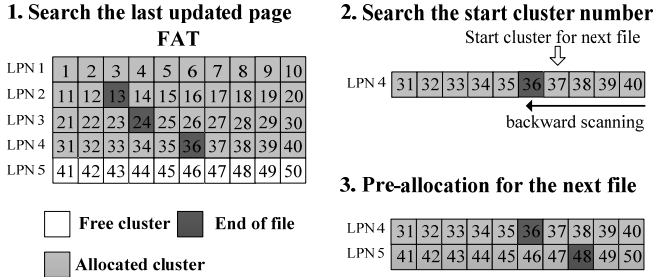


Fig. 4. Searching the start cluster number of unused cluster chain.

C. Flash-Aware Journaling

Although the proposed PCA and PAPA techniques can provide a high performance and high reliable file system by minimizing the number of write requests on storage and the number of metadata updates, these techniques cannot prevent the file system corruptions completely. If there is a system failure during metadata update operations, the file system consistency can be broken and some files can be lost. In order to solve this problem, a journaling technique should be adopted, which records the metadata changes at the journal area before the original metadata are updated. Then, using redo or undo operations, the file system can be recovered.

Several journaling techniques have been proposed for the FAT file system. The previous techniques maintain duplicated FATs for journaling. Only after the FAT in the journal area is updated, the original FAT can be updated. If there is a system failure while the original FAT is updated, it can be recovered with the duplicated FAT in the journal area. However, the duplicated FAT approach increases the overhead during file creation. Considering that general car DVR systems are designed only for simple video recording and the SD card provides a limited I/O bandwidth, the duplicated FAT approach will be unacceptable to car DVR systems. In addition, the scheme will exhaust the lifetime of SD card with the duplicated write operations.

The directory entries are also important metadata of the FAT file system. The previous journaling techniques write the DE update logs in the journal area. However, the techniques did not consider the characteristics of NAND flash memory. Whereas the size of a DE is 32 bytes, the page size of NAND flash memory is 4 KB or 8 KB. Therefore, a NAND flash memory page can store up to 256 DEs. Even though only one DE is modified, the whole page should be programmed. While a page that has several DEs is programmed, if there is a

program error, all the DEs in the page can be corrupted. Therefore, the journaling scheme should back up other unmodified DEs in the target page as well as the modified DEs.

Considering the limitations of the previous journaling techniques, this paper proposes a *flash-aware journaling* (FAJ) scheme. The FAJ scheme writes the page-level modification logs of FAT and DEs in the journal area. Since it does not maintain the duplicated FAT, the write traffic on the journal area can be minimized. In addition, using the page-level logging, the file system consistency is guaranteed even if there is a program failure.

Fig. 5 shows the overall journal structure of FAJ. The reserved area of FAT file system is used to store the journal data. The reserved area is hidden to user and it is originally used for vender-specific data. The size of reserved area is configurable. The FAJ scheme creates five DE journal areas and five FAT journal areas for five types of files at the reserved area. The total size of journal area is 640 KB.

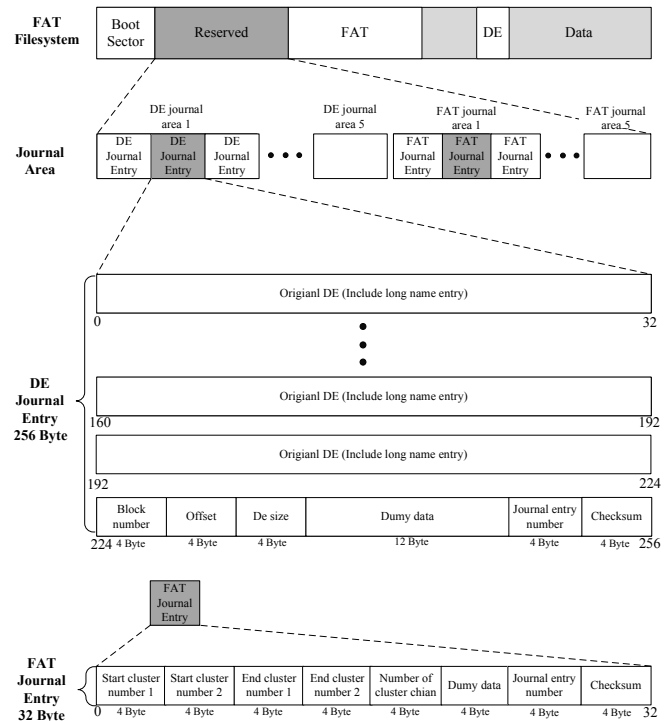


Fig. 5. Structure of the flash-aware journal.

Since an 8KB of flash page can contain 256 DEs, each DE journal area should contain 256 DE journal entries in order to cope with program errors. Even when only one DE is modified, FAJ stores all the DEs that share a flash memory page with the modified DE. Then, FAJ can recover the corrupted DEs even when a whole flash memory page is corrupted by program errors. Each FAT journal area should contain 2048 FAT journal entries since a page can have 2048 FAT chains at maximum. Each journal area is cleared when the original metadata are successfully written at the storage.

A journal entry has the metadata log of one file. The DE journal entry includes a copy of the original DE to be modified. It also has the original location information such as

the block number, the offset, and the size of DE. Since the FAT file system supports the long name, the size of DE is variable. The journal entry includes the journal entry index and the checksum also. The checksum is used to check the validity of journal entry.

The FAT journal entry includes the start/end cluster numbers and the size of the cluster chain. Due to the PCA scheme, a file is allocated with non-fragmented contiguous clusters. Therefore, the cluster chain can be represented with only the start and end cluster numbers. Since the cluster chain can be divided into two separated regions only if it is allocated across the partitioned region boundary, the FAT log has the second start/end cluster numbers also. Compared with the previous duplicated FAT schemes, the FAT journal entry has only minimal logs.

Under the page-aligned pre-allocation technique, the file recording is performed in three steps. First, the metadata is updated to create a file. Second, the buffered data are flushed. During the buffer flushing, there are no metadata updates at the proposed scheme. Finally, the metadata is updated again to de-allocate unused clusters at file close. The proposed journaling technique writes the metadata journals just before a file is closed as shown in Fig. 6. Therefore, if there is a failure during the first metadata update, the metadata cannot be recovered by the FAJ technique. However, this has no problem since no data are stored. Even if there is a system failure during the data flushing, the file system can access all the successfully written data due to the cluster pre-allocation. If there is a failure during journal write, the recorded file can be accessed with the metadata written at file creation. Finally, even if the metadata is corrupted during file close, it can be recovered by redoing the journal logs.

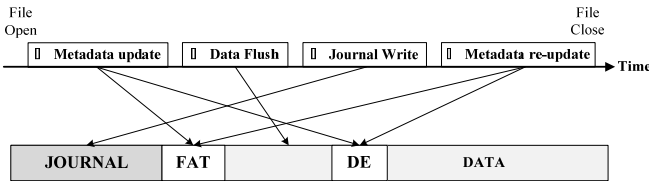


Fig. 6. Data and metadata accesses while recording a file.

IV. EXPERIMENTS

In order to demonstrate the effectiveness of the proposed techniques, several experiments are performed at a car DVR simulator and a real development board. The car DVR simulator receives the file system configuration and the workload pattern as inputs, and reports the file fragmentations and the storage internal behaviors as outputs. The car DVR workload pattern is collected from a real car DVR product. TABLE I shows the detailed configuration of the car DVR simulator. Three different time durations are used, during which the car DVR records five different types of data without formatting the file system. The storage spaces and the numbers of files for five types of data are based on the real workload. Although there are no accidents on the real workload, several driving events and parking events are generated due to the false negative detection by the car DVR system while crossing speed bumps and closing car door.

TABLE II shows the detailed specification of the car DVR development board, which uses one channel video and a class 10 SD card. The storage capacity, cluster size, and file size are same with those of car DVR simulator. At every 3 seconds, the buffered data are flushed at the SD card.

TABLE I
CAR DVR SIMULATOR CONFIGURATION

Duration	30 / 90 / 180 days				
SD Capacity	32GB				
Cluster Size	32KB				
Video Channel	1 Ch (74MB / min)				
Storage Space	Driving 65%	Driving Event 12%	Parking Motion 15%	Parking Event 5%	Manual 3%
Number of Files / Day	Driving 127	Driving Event 6	Parking Motion 6	Parking Event 1.3	Manual 0.3
File Size	Driving 74MB	Driving Event 24MB	Parking Motion 24MB	Parking Event 24MB	Manual 74MB

TABLE II
CAR DVR BOARD SPECIFICATION

CPU	600 MHz RISC
RAM	256MB DDR2 memory
Storage	256MB NAND Flash
Camera	Full HD
Slot	Support Micro SD card
Kernel	Linux kernel 2.6.35
Flush Interval	3 Seconds

A. Partitioned Cluster Allocation

With the car DVR simulator, the amount of file fragmentations is measured under different cluster allocation techniques. The original FAT file system suffers from the fragmentation problem. Fig. 7(a) shows the ratios of fragmented files for each file type under the original FAT file system. The number of fragmented files increases as the car DVR is used during a long time without formatting the SD card. Most of the files are fragmented after 180 days. Fig. 7(b) shows the average number of fragments of a file. One file is split into 11 regions on average and 21 regions at maximum. However, the PCA technique generates at maximum two fragmented cluster regions irrespective of the time duration. The maximum case is when the clusters of a file are allocated across the partition boundary.

The fragmented clusters will degrade the write performance. Fig. 8 demonstrates the performance degradations by the file fragmentation. Fig. 8(a) shows the latencies of file copy operation when free clusters are fragmented at several regions. The results are normalized by the latencies of a non-fragmented file. The latencies are measured at the car DVR development board. When the copied file should be written at 11 fragmented regions, the copy time is increased by 60% compared to the non-fragmented file since the file system should generate many small and random write requests. The copy time is increased up to by 2.4 times when the file is fragmented into 21 regions. Fig. 8(b) shows the numbers of flash memory block erases at

different cluster allocation techniques and different FTL schemes. Since the internal behaviors of SD card cannot be observed at a real system, the car DVR simulator is used. The simulator includes the BAST and FAST FTL simulators. The number of block erases increases as the time elapses. The original FAT file system shows more significant increases in the block erase count. Since the write requests on the fragmented file are small and random, the garbage collections of FTL are frequently invoked and the write amplification in the flash memory becomes significant. The block erase counts under the original FAT file system become more than two times the block erase counts under the PCA scheme after 180 days are elapsed. The lifetime of SD card is determined by the number of erase operations. Therefore, it can be known that the PCA scheme is effective to expand the lifetime of SD card.

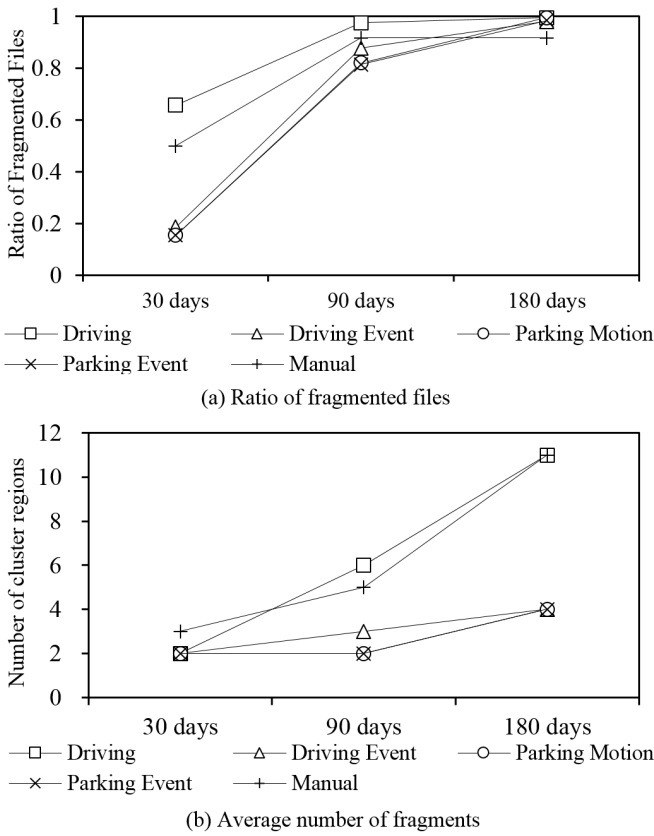


Fig. 7. Fragmentations of normal FAT file system.

B. Page-aligned Pre-allocation

The page-aligned pre-allocation technique has two advantages. First, it can reduce the number of metadata updates by allocating all the clusters at file creation. Second, it can reduce the number of overwrite operations on flash memory pages. In order to evaluate the effects of PAPA, the car DVR development board is used. TABLE III compares the number of metadata updates and the metadata update latencies under different cluster allocation techniques. The update latency consists of file system overhead, I/O scheduler overhead, and storage latency. The file system overhead includes the time taken to search free clusters from the FAT.

The I/O scheduler overhead represents the waiting time of write requests before they are sent to the storage device. The storage latency is the response time of storage device.

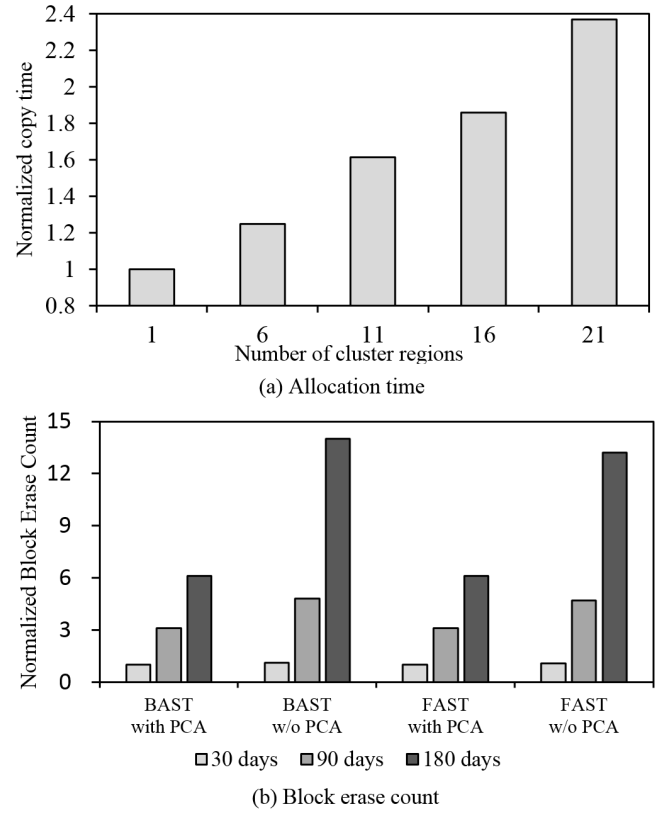


Fig. 8. Performance degradation by fragmentations.

TABLE III TOTAL METADATA UPDATE OVERHEAD FOR ONE FILE RECORDING			
		Original FAT FS	Page-aligned Pre-allocation
FAT	Number of updates	20	2
	File system (ms)	73.1	42.2
	I/O scheduler (ms)	88.1	7.1
	Storage (ms)	183.6	11.4
DE	Number of updates	20	2
	File system (usec)	740	82
	I/O Scheduler (ms)	393	0.8
	Storage (ms)	129.7	7.2

Since the car DVR system flushes the buffered data at every 3 seconds, the number of metadata updates is 20 in the original FAT file system. However, the PAPA modifies the metadata only when the file is created and it is closed. Although both the original file system and the PAPA technique allocate the same number of clusters for the target file, PAPA shows shorter file system overheads compared with the original scheme. This is because the cluster search operations are not required at the PCA technique. The I/O scheduler overhead and storage latency are also decreased by the PAPA technique since the number of write requests is reduced.

The PAPA technique reduces not only the number of metadata updates but also the probability of metadata corruption. Since FAT and DEs are updated infrequently and the update latencies are short, the failure-sensitive time intervals are also reduced.

TABLE IV
THE AVERAGE NUMBER OF FAT PAGE OVERWRITES

Schemes	The number of overwrites
Normal Pre-allocation	2.6
Page-aligned Pre-allocation	1.8

TABLE IV shows the number of overwrites at the pages allocated for FAT under different pre-allocation techniques. Whereas the normal pre-allocation scheme pre-allocates the clusters only for the current file, the page-aligned pre-allocation scheme pre-allocates the clusters for the next files also in order to update the FAT in the unit of 8KB page.

While recording ten 74 MB of files at SD card, the number of overwrites on the FAT pages is measured. The pre-allocation size is 75 MB. Therefore, several clusters can be de-allocated and the FAT is also modified at file close. With the page-aligned approach of PAPA, the number of page overwrites is reduced.

C. Flash-Aware Journaling

The failure detection and recovery operations of the proposed FAJ scheme are evaluated at the car DVR development board. The car DVR performs four steps of file recording as explained in Fig. 6. For experiments, the file system metadata or journals are corrupted manually at each step. When there are metadata corruptions, the file system consistency will be broken if the FAJ scheme cannot recover the metadata. The size of corrupted metadata ranges from a few bits to one logical page. If a whole page is corrupted, the metadata of previously created files can be corrupted as well as the metadata of the current file.

All the experiments showed that all the recorded files are successfully accessed despite corrupted metadata due to the recovery operation of FAJ.

V. CONCLUSION

The current FAT file system adopted by car DVRs has performance and reliability problems. To solve these problems, this paper proposed three file system techniques. The partitioned cluster allocation can improve the write performance and lifetime of SD card by removing fragmentations. The page-aligned pre-allocation can reduce the number of metadata updates and the probability of data corruption. The flash-aware journaling guarantees the file system consistency even when metadata is corrupted by system failures. The experiment results showed that these techniques improve the performance and reliability of the car DVR storage system.

This paper focused on the FAT file system. Recently, the car DVR systems begin to adopt a new file system called

exFAT in order to support a large capacity of storage. Since the exFAT file system also has similar performance and reliability problems, the proposed file system techniques will be applied to the exFAT file system.

REFERENCES

- [1] B. Carrier, *File System Forensic Analysis*, Addison-Wesley Professional, 2005, pp. 129-217.
- [2] R. Bez, E. Camerlenghi, A. Modelli, and A. Visconti, "Introduction to flash memory," in *Proc. IEEE*, vol. 91, no. 4, pp. 489-502, Apr. 2003.
- [3] M. Zheng, J. Tucek, F. Qin, and M. Lillibridge, "Understanding the robustness of SSDs under power fault," in *Proc. USENIX Conference on File and Storage Technologies*, pp. 271-284, Feb. 2013.
- [4] G. D. Nijs, A. Biesheuvel, A. Denissen, and N. Lambert, "The effects of filesystem fragmentation," in *Proc. Linux Symposium*, vol. 1, no. 1, pp. 193-208, Jul. 2006.
- [5] B. Ha, H. Cho, and Y. I. Eom, "A study on the block fragmentation problem of SSD based on NAND flash memory," in *Proc. 5th International Conference on Ubiquitous Information Management and Communication*, Feb. 2011.
- [6] Y. Kim and D. Shin, "High performance and high reliable file system for car digital video recorders," in *Proc. IEEE International Conference on Consumer Electronics*, pp. 193-194, Jan. 2015.
- [7] S. Park and S. Ohm, "New techniques for real-time FAT file system in mobile multimedia devices," *IEEE Trans. on Consumer Electronics*, vol. 52, no. 1, pp. 1-9, Feb. 2006.
- [8] S. Ryu, C. Lee, S. Yoo, and S. Seo, "Flash-aware cluster allocation method based on filename extension for FAT file system," in *Proc. 2010 ACM Symposium on Applied Computing*, pp. 502-509, Mar. 2010.
- [9] N. Kim and Y. Yu, "HFAT: log-based FAT file system using dynamic allocation method," *Journal of Information and Communication Convergence Engineering*, vol. 10, no. 4, pp. 405-410, Dec. 2012.
- [10] L. Alei, L. Kejia, L. Xiaoyong, and G. Haibing, "FATTY: a reliable FAT file system," in *Proc. 10th Euromicro Conference on Digital System Design*, pp. 390-395, Aug. 2007.
- [11] Y. Ryu, "A flash translation layer for NAND flash-based multimedia storage devices," *IEEE Trans. on Multimedia*, vol. 13, no. 3, pp. 563-572, Jun. 2011.
- [12] H. Kim, J. Kim, S. Choi, H. Jung, and J. Jung, "A page padding method for fragmented flash storage," in *Proc. International Conference on Computational Science and Its Applications*, pp. 164-177, Aug. 2007.
- [13] J. Kim, J. M. Kim, S. H. Noh, S. L. Min, and Y. Cho, "A space efficient flash translation layer for compact flash systems," *IEEE Trans. on Consumer Electronics*, vol. 48, no. 2, pp. 366-375, May. 2002.
- [14] S. Lee, D. Park, T. Chung, D. Lee, S. Park, and H. Song, "A log buffer-based flash translation layer using fully-associative sector translation," *ACM Trans. on Embedded Computing Systems*, vol. 6, no. 3, Jul. 2007.

BIOGRAPHIES



Younghun Kim received the B.S. degree in computer engineering from Sungkyunkwan University, Korea in 2014. He is currently a master student in the College of Information and Communication Engineering, Sungkyunkwan University. His research interests include embedded software, low-power, memory management, file systems and flash memory.



Dongkun Shin (M'08) received the BS degree in computer science and statistics, the MS degree in computer science, and the PhD degree in computer science and engineering from Seoul National University, Korea, in 1994, 2000, and 2004, respectively. He is currently an associate professor in the College of Information and Communication Engineering, Sungkyunkwan University (SKKU). Before joining SKKU in 2007, he was a senior engineer of Samsung Electronics Co., Korea. His research interests include embedded software, low-power systems, computer architecture, and real-time systems. He is a member of the IEEE.