Adapting Endurance and Performance Optimization Strategies of ExFAT file system to FAT file system for embedded storage devices

Keshava Munegowda [#1], Dr. G T Raju [#2], Veera Maninkandanraju [#3]

[#1]Principal Software Engineer,
Advanced Storage Division,
EMC Corporation
Bangalore, INDIA
keshava.gowda@gmail.com
keshava.munegowda@emc.com

[#2]Professor and Head,
Department of computer Science and Engineering, RNSIT,
Bangalore, INDIA
gtraju1990@yahoo.com

[#3]Senior Member Technical Staff,
Texas Instruments
Bangalore, INDIA
veera@ti.com

Abstract—The File Allocation Table (FAT) file system is commonly used in embedded storage devices such as Multimedia Cards (MMC) / Secure Digital (SD) / Micro SD cards, NOR, NAND flash memories. The Extend File Allocation Table (ExFAT) is the future file system for embedded storage devices. The MMC and SD card associations classify the ExFAT as the standard file system for storage flash cards of more than 32 Giga Bytes (GB) of size. This paper discourses the techniques to adapt the file read optimizations, Reduction of File Allocation Tables techniques and Cluster Heap for file write optimizations of ExFAT file system to FAT file system to improve the file read and write performance and the endurance of flash device

Keyword- Cluster, Contiguous, Directory, ECC, ExFAT, FAT, File system, Flash memories, MMC, Micro SD, NOR, NAND, NFAT, Storage

I. INTRODUCTION

The FAT [1] is widely used file system in tablet personal computers, mobile phones, digital cameras and other embedded devices for data storage and multi-media applications such as video imaging, audio/video playback and recording. The initial version of FAT file system was FAT12 by Microsoft Corporation, later it was extended as FAT16 and further as FAT32 to support higher storage capacity. The FAT file system was initially developed to use on floppy disks and hard-drives. Since most of the Personal Computer (PC) s implements the FAT file system, this file system has become a default and world-wide compatible storage format for embedded devices. Usually the device with the implementation of FAT is recognized as removable storage media in a PC. The Flash memories are default choice of any embedded device as they are low-priced, smaller size and higher storage capacity. Even though FAT file system does not define flash management techniques such as Wear-Levelling and Bad Block management, the embedded devices implements this file system along with the dedicated flash block management algorithms.

In FAT file system, the file or directory is the linked list of the clusters. A Cluster is a group of blocks or sectors of storage device. The File Allocation Table contains the linked list of clusters of files/directories. The FAT specification limits the maximum supported storage size to 32 GB. But, the maximum size supported by FAT32 implementation is 128 GB. But, today the flash storage cards of more than 32 GB are available in market. The ExFAT [2] [3] file system is developed, by Microsoft Corporation, as successor of FAT32 file system. This file system is optimally designed to support large size flash storage cards with higher read and write performance. The maximum storage size supported by ExFAT file system in 128 peta bytes (PB). The differentiating features of the ExFAT file system in comparison with FAT File system are

i) Reduction of instances of File Allocation Tables
ii) Clusters allocation optimizations for file write operation

iii) Contiguous clusters read algorithm [4]

The ExFAT file system is not compatible with FAT File system. But, it uses the clusters concept to store file/directory data and File Allocation Table to store the cluster chain of file/directory. This paper defines and implements the methodologies to adopt above techniques in FAT file system without breaking the compatibility of the existing FAT File system specification and windows and Linux operating system implementations. In addition, this paper defines the flash specific cluster allocation strategies applicable for both FAT and ExFAT File system.

II. Reduction of Instances of File Allocation Tables

In a FAT file system, there can be more than one instance of file Allocation Tables (by default 2 FAT instances). The reason for having multiple FAT instances is to provide redundancy for the FAT data structure so that if a sector goes bad in one of the FATs, that data is not lost because it is duplicated in the other FAT. This backup is useful for magnetic hard disks because of the high possibility of electric shock and vibrations of rotating disk platters and moving motor head used in magnetic hard disks affects the magnetic fields of the sectors and hence it corrupts the data of the sectors. On non-disk-based media such as flash devices, such redundancy is not necessary because flash memories do not use any mechanical rotating head. Moreover the block management software of flash memories uses the Error Correcting Codes (ECC) algorithms to correct the content of the block and vendor specific Bad Block Management technique to replace bad blocks.

As per FAT [1] specification, the field “BPB_NumFATs” of the boot sector or BIOS (Basic Input Output System) Parameter Block (BPB) indicates the instances of FAT. If there are many instances of FAT, for every file/directory creation, file write and file/directory deletion operation, FAT file system driver has to update all FAT instances. This will degrade the performance of these operations. Since erase and write are costly operations in flash devices, the multiple FAT instance update will lead to more garbage collection and in turn will reduce the life of the flash. Hence, The ExFAT file system uses only one File Allocation Table. The FAT file system can be used with single file allocation table. Following are the scenarios in which instances of FAT can be reduced to 1.

A. File system format operation

While formatting the device with FAT file system, the instances of FAT is written as 1. Figure 1 shows both conventional FAT file system organization and the same with single FAT. In Single FAT file system, the second instance of FAT is used as data clusters hence this configuration provides the more space to store user data along with the performance improvement. This approach is not applicable if the file system already contains user data because the format operation removes all files and directories of the file system.

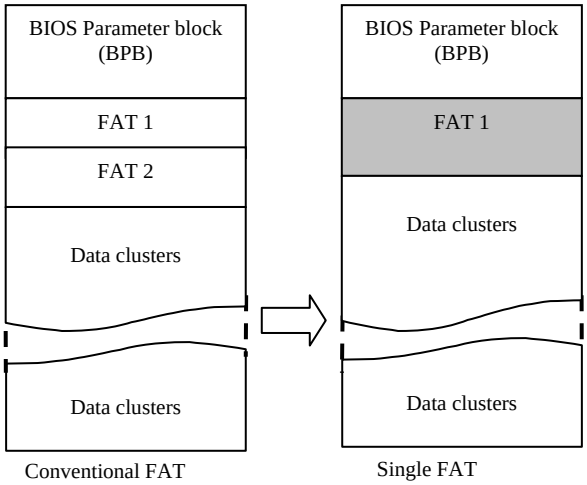

Fig. 1: Conventional FAT v/s FAT with Single File Allocation Table

B. Dynamic Reduction of FAT instances

This paper proposes the dynamic reduction of FAT instances during file system initialization or mount operation. If the flash storage is already formatted with two (or more) instances of FAT, the instances of FAT can be reduced to 1 by making FAT2 as FAT1 and sectors occupied by FAT1 are augmented to the reserved sectors count. Figure 2 depicts such FAT file system configuration. This approach does not remove the existing files/directories of the file system.

The existing mathematical formula [1] to calculate the first data cluster is as follows.

$$\text{RootDirSectors} \equiv \frac{((\text{BPB\_RootEntCnt} \times 32) + (\text{BPB\_BytsPerSec} - 1))}{\text{BPB\_BytsPerSec}}$$

```
If (BPB_FATSz16 != 0)
FATSz = BPB_FATSz16;
else
        FATSz = BPB_FATSz32;
```

$$\text{FirstDataSector} \equiv \text{BPB\_RsvdSecCnt} + (\text{FATz} \times \text{BPB\_NumFATs}) + \textit{RootDirSectors}$$

After adopting single FAT system, the calculation to determine the first data cluster is as follows:

$$\text{RootDirSectors} \equiv \frac{((\text{BPB\_RootEntCnt} \times 32) + (\text{BPB\_BytsPerSec} - 1))}{\text{BPB\_BytsPerSec}}$$

$$\text{BPB\_RsvdSecCnt} \equiv \text{BPB\_RsvdSecCnt} + (\text{FATSz} \times (\text{BPB\_NumFATs} - 1))$$

After incrementing the value of BPB_RsvdSecCnt, the value of BPB_NumFATs is set as 1.

```
                                        If (BPB_FATSz16 != 0)
        FATSz = BPB_FATSz16;
else
        FATSz = BPB_FATSz32;
```

$$\text{FirstDataSector} \equiv \text{BPB\_RsvdSecCnt} + \textit{FATSz} + \textit{RootDirSectors}$$

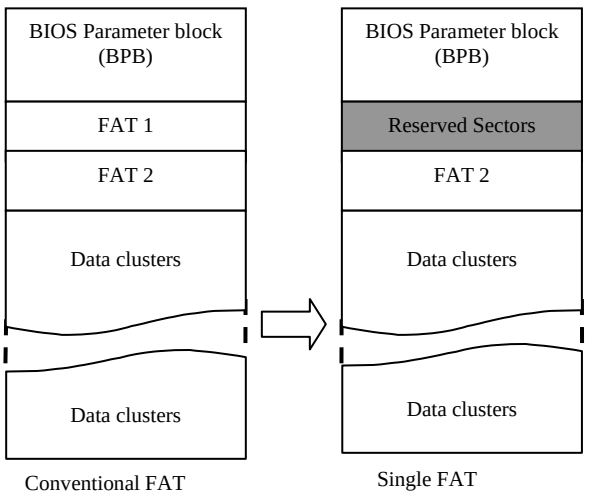

Fig.2. Conventional FAT v/s FAT with Single File Allocation Table with reserved sectors

By reducing the instances of FAT, the performance will be improved during operations such as file creation, file write and file deletion. During these operations, number of blocks write will get reduced. This will result in less number of garbage collection iterations, less erase operations and thus increases life of flash device.

III. CLUSTERS ALLOCATION OPTIMIZATIONS FOR FILE WRITE OPERATION

During file/directory creation and write operations, both FAT and ExFAT file systems searches for new cluster in file allocation table and allocated to a file/directory. But, The FAT and ExFAT file system uses a different approach for searching free clusters in the storage device. The Conventional FAT file system uses only the linked cluster allocation scheme whereas ExFAT uses linked cluster allocation scheme and contiguous cluster allocation scheme depending on the availability of the free clusters. The FAT file system examines the status FAT entries and if the status of the entry in the File allocation table is indicating free then the file data is written into the corresponding cluster and the FAT entry will be updated with allocation status. In case of FAT32 file systems, the free cluster search is performed starting from the cluster number specified in the field "FSI_Nxt_Free" (offset 492) [1] of the boot sector.

The ExFAT file system optimizes the free cluster search by using "cluster heap". The cluster heap is group of bits to indicate the status of all clusters of file system. Every bit in the cluster heap specifies the status of the data cluster, thus every byte indicates the status of 8 clusters. The binary value "0" indicates that cluster is free and value 1 indicates that cluster is allocated. The cluster heap is also referred as "Allocation Bitmap". The cluster heap is similar to block bitmap and inode bit map structures used in Ext2 [5] and Ext3 [6] [7] file systems. Figure 3 shows the logical organization of ExFAT file system and the structure of cluster heap. Usually Cluster Heap exists in the 2nd cluster of ExFAT file system.

During, file/directory creation and write operations, instead of searching 32 bit entries of File Allocation Table, the ExFAT file system search for the free cluster in the cluster heap; Since the cluster heap is smaller in size compared to File Allocation Table, it can be cached in the primary memory and the searching is optimal thus improves performance of the file write. In addition to free clusters search optimizations the cluster heap improves the performance of file write and delete operations. If a file/directory is allocated with the clusters which are contiguous then the "NoFatChain" [2] [3] bit field of generic secondary flags of the stream extension directory entry is set and status of the allocated clusters are set in cluster heap. The cluster chain will not be created in the File allocation table. This avoids the writes to file allocation table and hence it improves the file write performance. Similarly, whenever the file containing the contiguous clusters deleted, clusters status in the cluster heap are marked as free and no need to update the file allocation table thus it improves the file delete operation.The cluster scheduling method of NFAT [8] (New FAT file system) reduces the free cluster allocation time by using cluster scheduling algorithm operating in a cluster bank area and thus improves the file write performance, but this approach is not compatible with FAT file system. The backward search method of embedded FAT file system [9] always starts the search from tail of clusters chain assuming the availability of contiguous free clusters to allocate to files during write operation, but in this method searching for free clusters in non-empty file system requires longer time than conventional free cluster search method starting from the cluster number specified in the field "FSI_Nxt_Free" [1] of the boot sector.

The cluster heap structure of ExFAT file system can be implemented in FAT file system as a file containing the bit map for every cluster status such as allocated or free. In FAT file system, even though cluster heap can be placed as file, but to provide the backward compatibility it is required to update the file allocation table whenever the cluster is allocated or removed from the file. Hence, storing the cluster heap in a file causes additional writes to storage system whenever the cluster is allocated and removed from file/directory. To avoid, the additional disk writes, this paper implements the cluster heap for FAT file system in main memory. The status of each cluster is determined by traversing the entire File allocation table during file system initialization. The cluster heap implementation in FAT file system is limited only to optimize the free clusters search and it does not improve the performance of file delete operation.

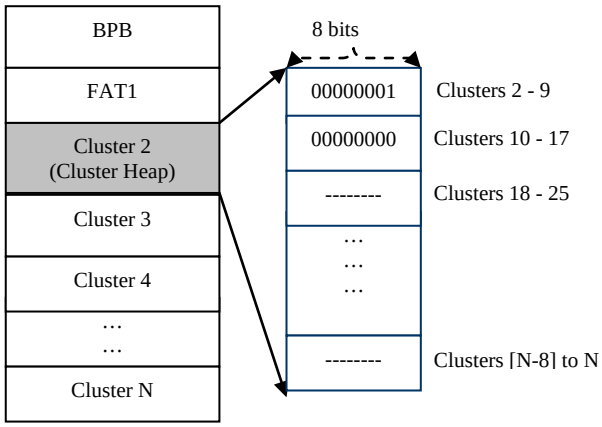

Fig.3. Cluster Heap in ExFAT File system

IV. CONTIGUOUS CLUSTERS READ ALGORITHM

In FAT file system, the file read operation consists of reading the content of data clusters and reading the cluster chain in FAT entries. Prior to file read operation , if the number of contiguous clusters of a file is known, then FAT entry read operation can be avoided to improve the file read performance. In ExFAT file system, the "NoFatChain" [2] [3] bit field of generic secondary flags of the stream extension directory entry indicates that all clusters of a file/directory are allocated in one contiguous series. If this bit is set, then FAT entry read operation is not required while reading the file. But, this method gives the best performance only if all the clusters of a file are allocated in a one contiguous series. If there are multiple groups of contiguous clusters allocated to a single file then FAT entry read operations are required during file read. Moreover, the stream extension directory entry cannot be used in FAT file system because the ExFAT file system is not compatible with FAT file system. In FAT32 file system, the maximum number of clusters allowed is 2^{28}-1. This is because, even though File Allocation Table is of 32 bit entry, File system uses only 28 Bits; the most significant 4 bits are not used. This paper implements the clusters group read algorithm which uses these 4 bits to store the signature value indicating that file is created/appended using contiguous clusters allocation algorithm. The 4bit signature value of the cluster entry at first cluster index of file allocation table identifies whether file is created in the directory/small files segment or larger files segment. Following the signature value, the 16 bit value formed by the most significant 4 bits of next four cluster entries indicates the number contiguous clusters or set of clusters allocated to the file. This algorithm determines the number of contiguous clusters following the first cluster of the file by reading most significant 4 bits of 2^{nd}, 3^{rd}, 4^{th} and 5^{th} cluster FAT entries of the file; this reduces number of FAT entry read operations and hence improves the performance. Figure 4 illustrates that FILE1.TXT has the starting cluster number 3, the value of the most significant 4 bits of the FAT cluster index 3 is the signature value indicating that file is allocated with contiguous clusters. The 16 bit hexadecimal number formed by concatenating the most significant 4 bits of next 4 entries (cluster index 4, 5, 6, and 7) indicates the number of contiguous cluster allocated the FILE1.TXT.

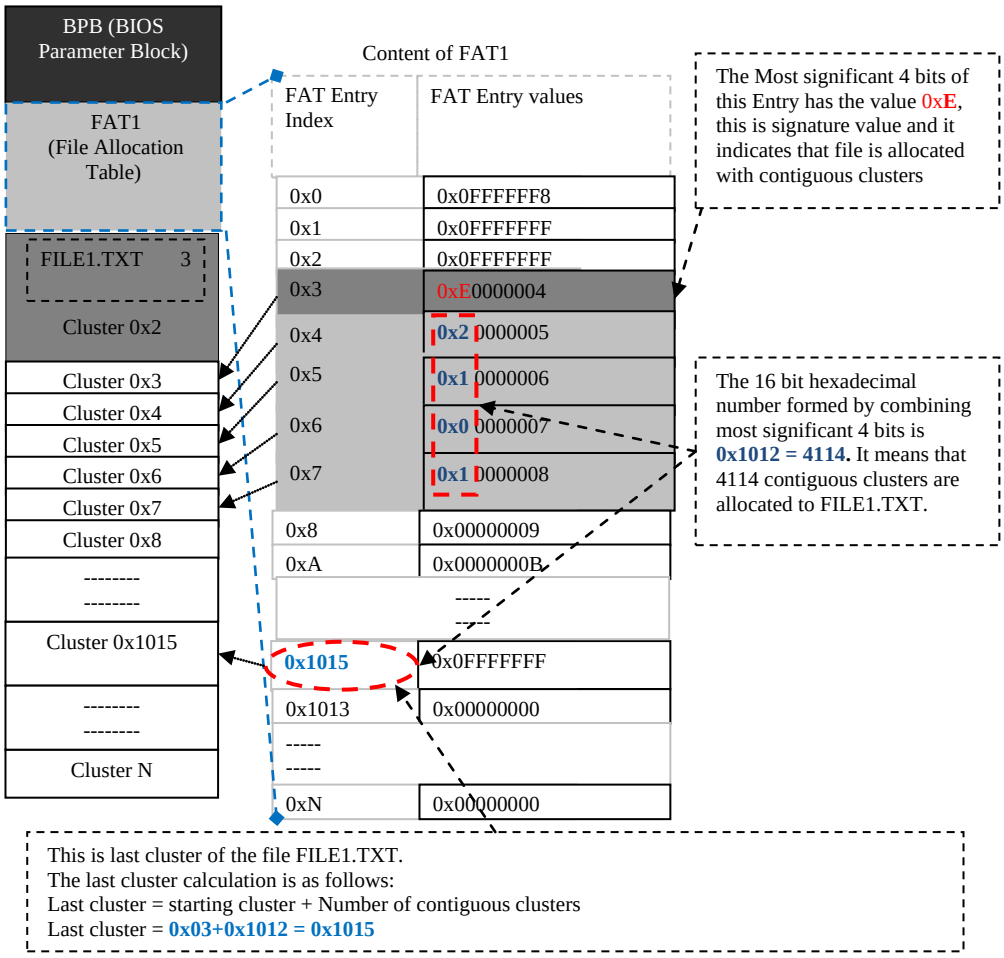

Fig. 4. FILE1.TXT File with contiguous clusters in directory/small files segment

The usage of most significant 4 bits of the FAT entry to indicate the contiguous clusters allocation is limited to FAT32 file system. In FAT12 and FAT16 file system, the complete 12 and 16 bits of FAT entry are used to

represent the cluster number of file/directory. The "DIR_NTRes" field of the 32-bytes directory entry structure of the file/directory can be used to indicate the contiguous clusters allocation. Even though, the FAT specification [1] indicates that this field is reserved and recommends the value 0 in this field, the Windows NT, XP and higher versions of the operating systems uses this field to maintain the English language alphabets case sensitivity while displaying the SFN (Short File Name) and case insensitivity while searching for SFN. If all 8 characters of the name part of SFN are lower case, then 4th bit of DIR_NTRes field will be set to value 1. Similarly, if all 3 characters of the extension part of SFN are lower case, the 5th bit of DIR_NTRes field will be set to 1. The bits other than 4th and 5th bits of DIR_NTRes field of the 32 bytes directory entry structure of the file/directory can be used indicate the allocation of contiguous clusters allocation to a file. The implementation this paper uses the 8th bit of DIR_NTRes field to indicate the allocation of contiguous clusters to file/directory. The usage of 8th bit of DIR_NTRes is applicable to FAT32 file system too, but this approach works only if all the clusters of a file are allocated in a one contiguous series. If there are multiple disjoint groups of contiguous clusters allocated to a single file, then usage of most significant 4 bits of FAT entry indicates the number of contiguous clusters for every cluster group.

V. CONTIGUOUS CLUSTERS READ ALGORITHM

Typically, the Data is written to the flash memory in units called pages/sectors. Each page is made up of multiple cells. Each memory cell stores single bit information (either 0 or 1). However, the memory can only be erased in larger units called blocks (made up of multiple pages) [10]. The maximum cluster size in FAT file system is 64KB. The maximum cluster size in ExFAT file system is 32MB. Both the file systems, determines the cluster size to be used depending on the total number of sectors/pages in the storage device. This type of cluster allocation leads to a situation where pages/sectors of single erase unit block is allocated to different files or directories. Upon deletion of a file or directory, the garbage collection algorithm needs to be executed to identify the stale data and the valid data in a block. In case of wear-leveling, the valid data is moved to different block so that block containing the stale data is erased. This reduces the performance of the file write and deletes operations. The flash specific cluster allocation technique determines the cluster size based on the block size / erase-unit size. The block size of the MMC / SD card can be determined by reading the SECTOR_SIZE [11] field of the Card Specific Data register (CSD) of the MMC/SD card device. For FAT32 file systems, if the erase unit size is less than 64KB, then erase unit size is used as cluster size of the file system. If the erase unit size is more than 64KB, then cluster size is limited to 64KB only. Similarly, for ExFAT file system, if the erase unit size is less than 32MB (Mega Bytes), then erase unit size is used as cluster size of the file system. If the erase unit size is more than 32MB, then cluster size is limited to 32MB only. This technique allocates all the pages/sectors of single block to a single file or directory. Due to this, upon file or directory deletion , the complete content of the erase unit block will be marked as stale data and same block will available for wear-leveling and there is no need execute the garbage collection and hence improves the file system performance.

VI. EXPERIMENTAL RESULTS

The Performance Benchmarking of file write and read operations are conducted on Dell laptop named Inspiron 5520 containing 4 Intel core i5 processors of 2.5 GHz speed and 4GB of RAM. The file system benchmarking tool IOzone version 3.3 [12] [13] is used in the Linux kernel based Ubuntu version 12.04 operating system for the performance measurements. The IOzone software uses the record size of 4MB (Mega Bytes) for file read and writes operations. File sizes used for the performance benchmarking are 32MB to 512 MB. Both ExFAT and FAT32 file systems sets the cluster size as 64KB (Kilo Bytes) for USB thumb drives. For MMC/SD cards, the implementation of this paper read the erase unit size from SECTOR_SIZE [11] field of the Card Specific Data register (CSD) of the MMC/SD card driver while mounting the file system. The implementation of optimized cluster allocations and cluster group read algorithm modifies the FAT file system code of the Linux kernel version 3.5.7 used in Ubuntu 12.04 operating system and it is denoted as "Modified FAT". The Linux FUSE (File system in User Space) [14] based ExFAT implementation [15] is ported to Linux kernel to achieve the accuracy in the performance benchmarking. The SanDisk 64GB SD card with cluster size of 64KB is used for the performance benchmarking and it is mounted with buffer cache disabled in the Linux kernel. Figure 5 shows the file write performance results of conventional FAT32, Modified FAT32 and ExFAT file systems. The ExFAT yields the 2.5 times more performance improvement than FAT32 file system during write operations depending on firmware of the memory card. The Modified FAT file system improves the write performance by 70% to 90%. The average write performance of FAT32, Modified FAT and ExFAT file systems are 2.7 MB/S (Mega Bytes per Second), 4.9MB/S and 7.4 MB/S respectively. Figure 6 shows the file read performance of the FAT32, Modified FAT and ExFAT file systems. Compare to FAT32 file systems, Both ExFAT and Modified ExFAT file systems yields the performance improvement of 2 MB/S in file read operations. The ExFAT file system provides the negligible performance improvement over Modified FAT file system in file read operation. If there are larger file reads, the ExFAT and Modified FAT file systems yields more performance improvements. In case of 64 GB SanDisk SD card, the average read performance of FAT32, Modified FAT and ExFAT file systems are 29.3 MB/S, 32.4 MB/S and 32.7 MB/S respectively.

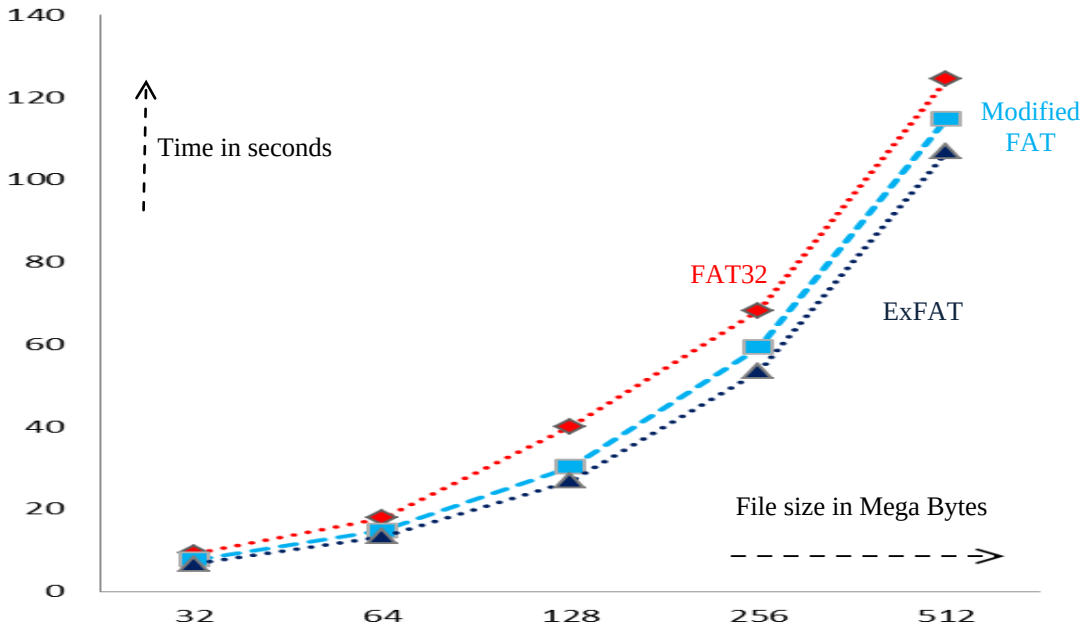

Fig. 5. File write performance comparison between FAT32, Modified FAT and ExFAT File systems

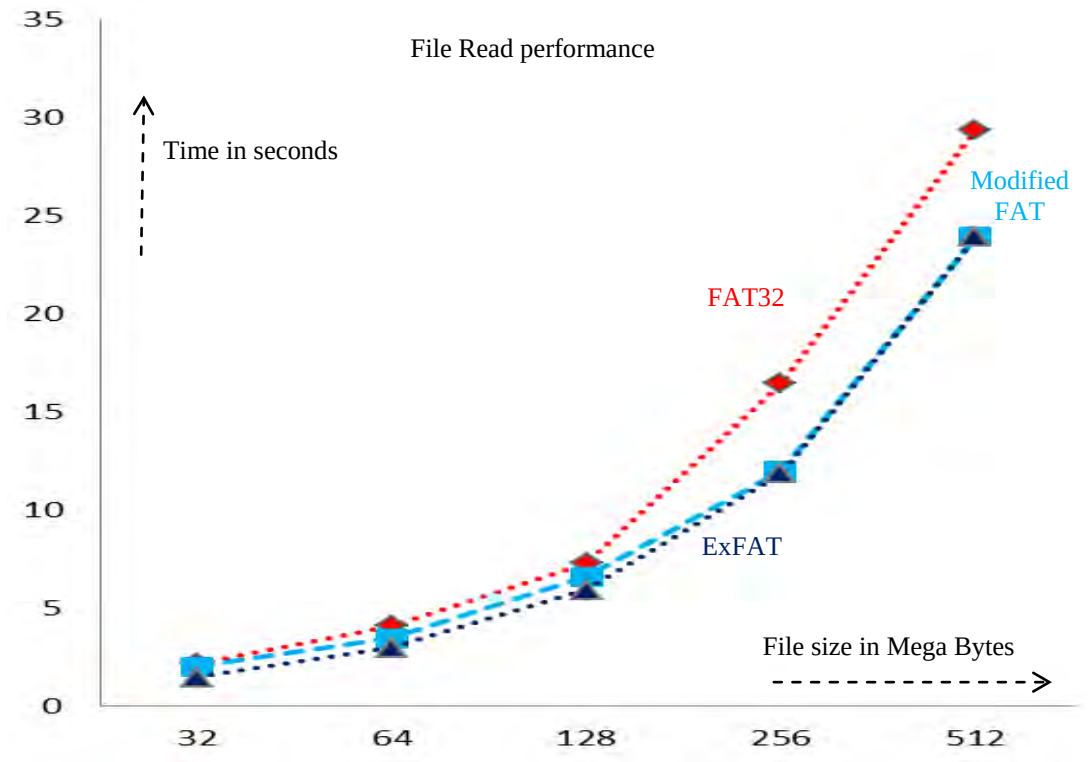

Fig. 6. File read performance comparison between FAT32, Modified FAT and ExFAT File systems

VII. CONCLUSION

The File Allocation Tables Reduction and Cluster Allocation techniques provide high write performance improvement. The FAT reduction minimizes the metadata updates per file/directory hence it improves the endurance of the flash device. The Dynamic FAT reduction technique yields the free sectors between the boot sector and File allocation Table of the file system. These sectors can be used for file/directory metadata journaling during file write operations to provide power fail safe feature. The file read optimization yields more performance for larger size files.

ACKNOWLEDGMENT

It is a pleasure to acknowledge Mr. Robert Shullich, Enterprise Security Architect, Tower Group Companies, USA, for resolving technical queries to understand the ExFAT file system. Mr. Andrew Nayenko deserves special thanks for open souring the FUSE based ExFAT file system implementation to Ubuntu and other Linux operating system distributions. This paper owes great deal of thanks to Mr. Madan Srinivas and other members of storage team of Texas Instruments (India) Pvt Ltd and Samsung (India) Pvt Ltd for helping us in porting the FUSE based ExFAT implementation as a part of Linux kernel.

REFERENCES

[1] FAT32 File System Specification, FAT: General Overview of On-Disk Format, Microsoft, 2000.
[2] Ravishankar V.Puipeddi, Vishal V.Ghotge, Ravinder S.Thind, “Quick file name look up using name hash”, USPTO Patent: 8321439, granted on November 27, 2012.
[3] Keshava Munegowda, Venkatraman S, Dr. G T Raju, “The Extend FAT file system: Differentiating with FAT32 file system”, Linux Conference, Prague, Czech Republic, Europe, October 2011.
[4] Ravishankar V.Puipeddi, Vishal V.Ghotge, Ravinder S.Thind, “Contiguous File Allocation in Extensible File system”, USPTO Patent: 8,606,830, granted on November 20, 2013.
[5] Remy Card, Theodore Ts’o, Stephen Tweedie, “Design and implementation of the Second Extended File system”, First Dutch International Symposium on Linux
[6] Stephen C. Tweedie (May 1998). "Journaling the Linux ext2fs File System, Proceedings of the 4th Annual LinuxExpo, Durham, NC. Retrieved 2007-06-23
[7] Dr. Stephen C. Tweedie , “Ext3 , journaling file system”, Ottawa Linux Symposium, Ottawa Congress Centre, Ottawa, Ontario, Canada on the 20th of July, 2000
[8] Moonsoo Choi, Heemin Park, and Jaewook Jeon, “Design and Implementation of a FAT File System for Reduced Cluster Switching Overhead”, International Conference on Multimedia and Ubiquitous Engineering, 2008.
[9] Zhang Jinhai, “Research of Embedded FAT file system,” IEEE International conference on Uncertainty reasoning and knowledge engineering, 2011.
[10] Arnd Bergmann, “Working with Cheap Flash Drives”, Embedded Linux Conference, Sanfrancisco, USA, April 2010.
[11] SD Specifications Part 1: Physical Layer Simplified Specification version 4.10, SD card Association, January 22, 2013.
[12] Keshava Munegowda, Dr. G T Raju, Sourav Poddar, “FFSB and IOzone: File system Benchmarking Tools, Features and Internals”, Embedded Linux Conference, Barcelona, Spain, Europe, November 2012.
[13] IOzone, “File systems Benchmarking tool”, http://www.iozone.org
[14] FUSE (File System in User Space) : http://fuse.sourceforge.net/
[15] Fuse based ExFAT implementation for Linux : http://code.google.com/p/exfat/