A New Design of In-Memory File System based on File Virtual Address Framework

Fahad Samad
Department of Computer Science
FAST – National University of Computer and Emerging Sciences
Karachi, Pakistan

Zulfiqar Ali Memon
Department of Computer Science
FAST – National University of Computer and Emerging Sciences
Karachi, Pakistan

***Abstract*—Rapid growth in technology is increasing day by day that demands computer systems to work better, should be reliable and have faster performance with fair cost and best functionalities. In the modern era of technology, memory files are used to shorten the performance gap between memory and storage. Sustainable in-memory file system (SIMFS) was the first that introduces the concept of open file address space into the address space of the process and exploits the memory mapping hardware while accessing files. The purpose of designing and implementing the SIMFS architecture is to achieve performance improvement of in-memory file system. SCMFS are designed for the storage class system that uses the presented memory management component in the operating system to assist in managing block, and it manages the space for each and every file adjacent to the virtual address space. A recent study has proposed that non-volatile memories are powerful enough to minimize the performance gap, as compared to previous generation non-volatile memories. This is because the performance gap between non-volatile and volatile memories has been reduced and there are possibilities of using a non-volatile memory as a computer's main memory in near future. Lately, high-speed non-volatile storage media, such as Phase Change Memory (PCM) has come into view and it is expected that for storage device PCM will be used by replacing the hard disk in upcoming years. Moreover, the PCM is byte-addressable, it means that it can access individual byte of data rather than word and data access time is expected to be almost indistinguishable of DRAM, a volatile memory. These features and innovations in computer architecture are making the computer system more reliable and faster.**

Keywords—Phase change memory; non-volatile memory; Spin Transfer Torque – RAM; sustainable in-memory file system; journaling file system

I. Introduction

With the passage of time, increasing demands of new technologies with better and faster performance and rapid data processing at a reasonable cost, demand system to be designed accordingly to behave and work efficiently [1].

Many new designs are incorporated to fulfill these functionalities. Likewise, we have in-memory file systems that are used to diminish the performance gap between memory and storage device. There are two types of in-memory file system, i.e. temporary and persistent. The temporary file system may not capable of retaining metadata and data may not undergo on system reboots while in persistent in-memory file system we have insistent data that preserve on system reboots. The novel blueprint of in-memory file system has come up with Virtual file address for increasing performance of in-memory file system. Each individual file has a virtual address space which is controlled by a file page table [2].

Sustainable in-memory file system (SIMFS) is intended and executed on same structure mentioned above. This employs the memory mapping hardware while accessing files. And the data are also managed by file page table. The challenges in designing this framework is to create a persistent metadata and then designing this file system by incorporating file data to virtual address space. For persistent storage of data, non-volatile memories are used that are directly connected via the memory bus which reduces the latency and they are byte addressable as well [3]. The file system which builds on the virtual memory space exploits Memory Management Unit (MMU) to map the address of the file with virtual address. These features of non-volatile memory have replaced DRAM – volatile memories. But still, non-volatile memories are sheathed in performance, however, memory devices such as STT-MRAM (Spin Transfer Torque Random Access Memory) has overcome this problem as it has comparable read/write access time. They are capable enough to maintain the data of main memory into main memory still after system gets turned off. STT-RAM pursues all the characteristics of a universal memory.

Now a day, a wide gap between disk and main memory has become a severe drawback in the computer system. To overcome this, the operating system stores disk block which requests for data into some part of main memory which is called buffer cache. Buffer cache works even in conditions when storage is working faster in main memory. Moreover, Phase Change Memory (PCM) has appeared as new storage medium and expected to be used as main memory in the near future as well [4].

The reason for using PCM is because it's a non-volatile storage mechanism and has increased density and significant power consumption. It also has amplified performance when replacing DRAM with PCM. The Journling file system provides high dependability at rational cost, however, existing systems doesn't support a PCM storage as they are hard disk optimized. The new journaling file system is introduced for PCM, named JFS that cut off write traffic to PCM as compared to the existing journaling file system. JFS uses less data as compared to the existing journaling file system.

II. Background Study

Increased timing in data processing increases the performance gap between memory and storage. This probably leads to disadvantage of computing system as compared to modern systems [5]. Not only this, the existing file system is complicated to apply directly on memory. Moreover, the disk based file system uses volatile memories that are faster but have slow secondary storage as per traditional architecture, such as Linux. In addition, the I/O data request has to search bottomless stack of software layers [6].

These problems lead to the solution for making up file systems that are "in-memory" file systems. Some modern system uses this framework, like, SPARK, which cluster computing framework and uses in-memory file system to some extent.

According to research experiments, there is a vast performance difference between disk and main memory. File access by the disk is 5000 to 8000 times slower than the file access by in-memory file system.

Above framework can be described by using an example of the file read. An in-memory file only takes a minute to read a file, whereas, simple file using the existing system may take 4 days to read that file. The example, stated above clearly explains the advantage of the in-memory file system. They play a vital role to benefit applications involving in data processing. Our main goal is to focus on designing an in-memory file system that should be faster and persistent on file reads / writes. No matter either it is sequential access or random access.

Our main focus is on persistent in-memory file system. We have two types of in-memory file system, i.e. persistent and temporary. Temporary in-memory file system has no sustainable metadata and data is lost on power disconnection while persistent in-memory file system has sustainable metadata and can survive on system reboot. So this makes necessity to have persisted in-memory file system so that we can have sustainable metadata and storage as well [7].

In the existing system we were taking some things for granted and that leads to the huge performance gap between memory and storage.

By launching new framework with modern techniques we also come up with the term "File Virtual Address Space" [8].

Existing systems may not have file virtual address space for an individual file. This new framework introduces file with its own, virtual address space.

III. Research Work

With the goal of designing new in-memory file system, we came up with sustainable in-memory file system (SIMFS), which is designed and implemented in the same framework discussed above. Each opened file has its own, virtual address space that is represented by a hierarchical page table. For locating a physical location of the file system by the virtual address space of a file, a file system may use memory management unit (MMU). The above framework implements bye addressable memory that is connected to a memory bus [9].

The SIMFS architecture easily integrates virtual address space of file into virtual address space of processes and uses the same hardware MMU for this task. They are good and better than all those in-memory file system mentioned as an example in this paper.

When discussing about basic file systems and different kinds of data, we came to know about metadata and physical file data. The metadata store file attributes and they are mapped by logical location to the physical location of each file data page [10]. We termed this mapping as mapping structure (Fig. 1).

Fig. 1 shows mapping of metadata into data sections which is represented as pages or we can say block. The block diagram clearly explains the mapping structure of metadata. As far as existing systems are concerned, they may contain many of the file system.

Starting from the typical disk-Based system that includes EXT2 and EXT4 and both of them are inode structure. Ext4 is much more improved version that EXT3 and so on. Ext4 has the concept of in-memory file system. In the same way, we have persistent and non-persistent in-memory file system. Persistent file system may embrace Protected & Persistent RAM based file systems (PRAMFS) that are 2-D structured and are light weighted and also may have sufficient storage of space with non-volatile file system. Second, persistent file system includes Persistent Memory File System (PMFS), which are also light weighted and have B-Tree structure and are capable enough to provide access to persistent memory with the CPU directly through load/write instruction [11].

Whereas, non-persistent in-memory file system may consist of random Access Memory File System (RAMFS) and Temporary File System (TMPFS). RAMFS works on the same principle of in-memory file system with storage space. While, TMPFS is a modern RAM file system which overcomes the drawbacks of the RAMFS file system. It limits the size for disk and show disk full error of that function. This is a better way than RAMFS. Both the non-persistent memory described above is Radix tree structured. In all the above stated file system, there is no such essential change, all have to search metadata through software routines for finding the physical address of each data pages.

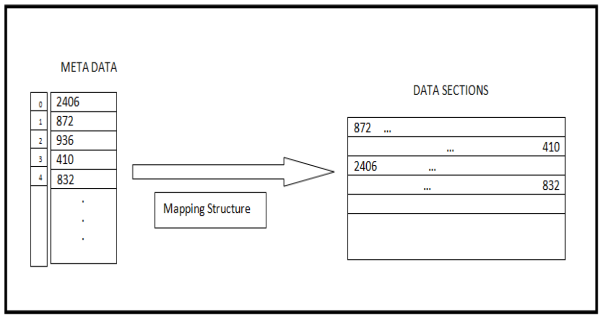


Fig. 1. A block diagram of metadata mapping in data section.

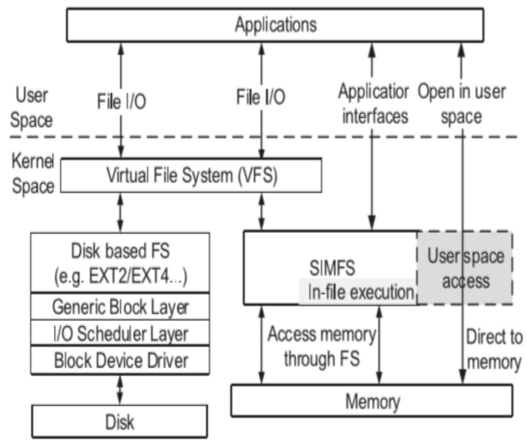


Fig. 2. A hierarchical view of multiple types of file systems.

The framework we are discussing, i.e. SIMFS doesn't follow the existing systems architecture instead it avoids software overheads which are used by existing file system. So here we use Memory Management Unit (MMU) to access file with virtual address space. This can help us to read any page without metadata searching in software as we were doing in existing systems. File virtual address space is represented by a file page table and it has exact similar structure as of process page table.

SIMFS perform the above task by placing pointers at the top level of the file page table in user's process.

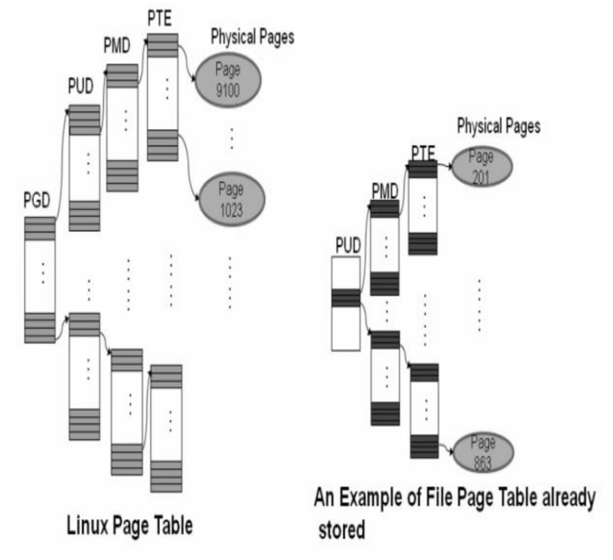
Through this, an application is capable to directly access file data by its own address space and no need to copy data in the user's buffer, as shown in Fig. 2

A new technique is proposed for application operation named as in-file execution. Previously used systems originates larger overhead in making data copies between files and buffers, but, SIMFS work on the principle which is entirely different from traditional methods and present new interface for applications shown in Fig. 2. Using in-file execution process, applications are independent enough to manage files in the file system and no now no need of copying data into buffers.

IV. Architecture, Design and Framework

The framework discussed above proposed the notion that each file has its own virtual address space. When an opened file is being processed, virtual address space of that file is entrenched into process's virtual address space. Therefore, each opened file has its own, virtual address space. By the help of "File page Table" physical space of the file is mapped with a virtual file space. When the file is closed, the virtual address space of the file is isolated from the virtual address space of the process.

The architecture proposed an effective way to access file via in-memory file system. To organize the virtual address space of a file, more efficiently file page table is used, that keeps the information about mapping address for each data page file. File page table is same as of process page table.



(a) (b)

Fig. 3. An organizational view of Linux page table.

The detailed working of this framework is shown in Fig. 3.

Fig. 3 illustrates the example of a file page table in Linux based operating system. Fig. 3(a) demonstrate Linux page table that contains four entries, i.e. PGD, PUD, PMD, and PTE. Each level in the page table stores the initial physical address of the page appearing next in the page table. For e.g. PUD level may store the pointer of a PMD physical page appearing next in the page table.

Fig. 3(b) illustrates an example of file page table that is already stored in the page table. They also contain three levels, i.e. PUD, PMD and PTE. They are similar as a Linux page table that each level in the page table stores the initial physical address of the page appearing next in the page table. For e.g. PMD level may store the pointer of a PTE physical page appearing next in the page table. The top-level of the page file table, i.e. PUD is accumulated in an inode structure of the equivalent file. As stated in Fig. 3(b), all the file data pages are arranged in an adjacent virtual address space, but actually they are dispersed on physical memory. Every individual file has a file page table within this framework.

When we open a file we simply insert file virtual address space into process virtual address space and it takes only O (1) time for inserting into virtual address space of the process. And it can be easily done by copying few pointers into the file page table at the highest level. After the insertion with an adjacent virtual address space of file, any location can be acquired easily in the file without searching of metadata through software routines (Fig. 4).

Fig. 5 clearly explains the mapping of virtual address space into physical pages of file via memory mapping hardware (MMU).

For example, Offset address 8000 of the files has been just equal to the beginning address + 8000. So this can foster the performance and it is especially fastest for random read/write access. This process quickly finds any location of the file.

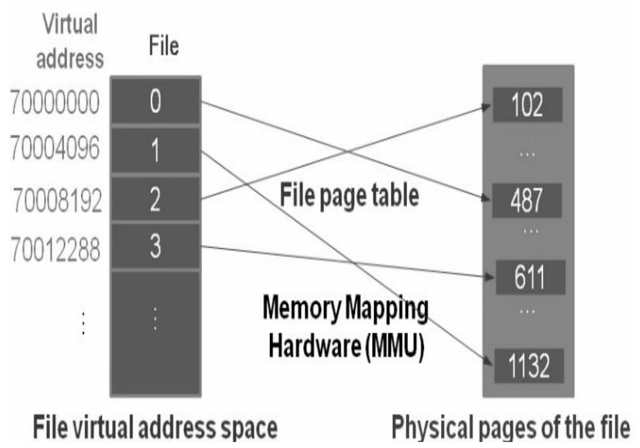
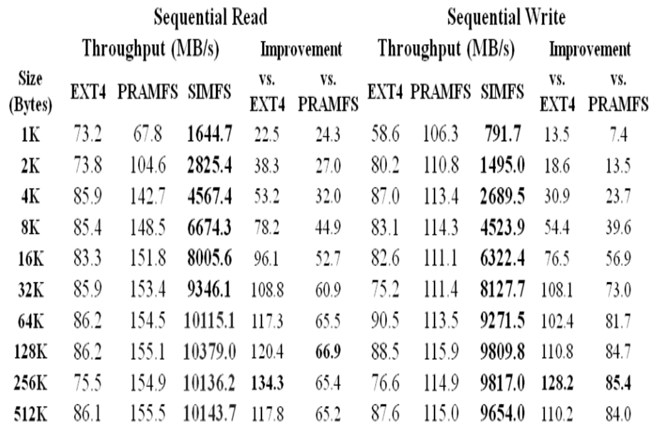


Fig. 4. An illustration of file virtual address space and physical pages of file.



	Sequential Read					Sequential Write				
	Throughput (MB/s)			Improvement		Throughput (MB/s)			Improvement	
Size (Bytes)	EXT4	PRAMFS	SIMFS	vs. EXT4	vs. PRAMFS	EXT4	PRAMFS	SIMFS	vs. EXT4	vs. PRAMFS
1K	73.2	67.8	**1644.7**	22.5	24.3	58.6	106.3	**791.7**	13.5	7.4
2K	73.8	104.6	**2825.4**	38.3	27.0	80.2	110.8	**1495.0**	18.6	13.5
4K	85.9	142.7	**4567.4**	53.2	32.0	87.0	113.4	**2689.5**	30.9	23.7
8K	85.4	148.5	**6674.3**	78.2	44.9	83.1	114.3	**4523.9**	54.4	39.6
16K	83.3	151.8	**8005.6**	96.1	52.7	82.6	111.1	**6322.4**	76.5	56.9
32K	85.9	153.4	**9346.1**	108.8	60.9	75.2	111.4	**8127.7**	108.1	73.0
64K	86.2	154.5	10115.1	117.3	65.5	90.5	113.5	**9271.5**	102.4	81.7
128K	86.2	155.1	10379.0	120.4	**66.9**	88.5	115.9	**9809.8**	110.8	84.7
256K	75.5	154.9	10136.2	**134.3**	65.4	76.6	114.9	**9817.0**	**128.2**	**85.4**
512K	86.1	155.5	10143.7	117.8	65.2	87.6	115.0	**9654.0**	110.2	84.0

Fig. 5. A table showing sequential read/write access of different in-memory file systems.

A. *Innovation in SIMFS Architecture*

It anticipated the concept of File virtual address space characterized by file page table build in the form of process page table.

Each file has its own contiguous virtual address space and no divergence among them. We can easily incorporate file virtual address space into process virtual address space swiftly, independent of file size. Moreover, file access takes benefit of the hardware Memory Mapping Unit in CPU.

V. Experiment and Comparison

SIMFS set an example of state of the art in-memory file systems with better performance and benefits as compared to traditional and existing file systems. SIMFS has also implemented functions that EXT4 has applied and comparing both of them SIMFS still have greater and better performance than EXT4 (Fig. 5).

The above figure clearly shows the best results of SIMFS than existing file systems. SIMFS directly move towards the bandwidth of the memory bus hardware.

At sequential read, SIMFS is 66 and 120 times faster than PRAMFS and EXT4 for the size of 256KB, respectively. On sequential writes, SIMFS is 85 and 128 times faster than PRAMFS and Ext4 for the size of 256KB, respectively.

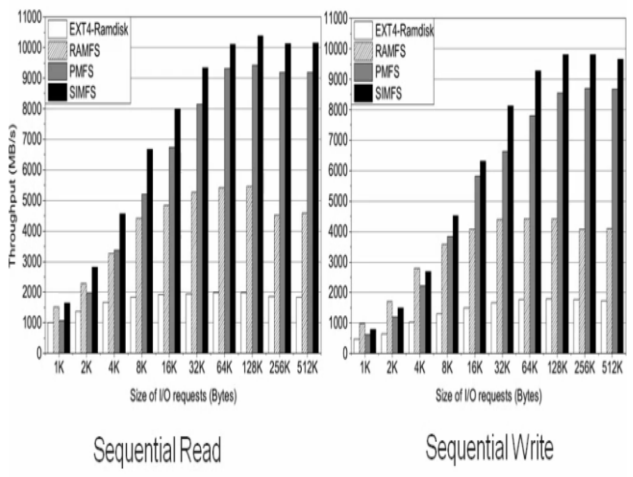


Fig. 6. Graphical comparison of SIMFS with different in-memory file system.

While comparing with other in-memory file system we have again SIMFS showing the best results among all of them. SIMFS is 4 times faster than EXT4 on ramdisk and on average results 2.2 and 2.1 times faster than RAMFS and PMFS, respectively. While the PMFS is the state of the art in-memory file system but still it lags in performance as compared to SIMFS framework (Fig. 6).

VI. Related Work

The promising technologies with persistent memories, including phase change memory (PCM), MRAM and many more have non-volatile memories. Non-volatile memories are used in almost every gadget including, laptops, mobile, tablets, flash memories, etc. Most of the devices in the past used only electrons that weren't reliable enough. Here, in non-volatile memories, we are using electronic conduction by modulating it by ion motion, moving oxygen ion one side to another or we can say to the upper level. From this technique we create a cell that can best select without a transistor and by doing this we can achieve 2D memory to 3D memory. These memories give us four times higher performance on almost the same cost with larger capacity, cheaper product and longer battery life. Non-volatile memories are widely used in systems as they have persistent data and survival of data on system reboots but volatile memories are vice versa of non-volatile memories. The data is lost when power is disconnected from the device.

A. *Integrating Memory Management*

Nonvolatile Memory (NVM) technologies came up with improved performance and capacity, faster access speed and with cheaper cost.

According to recent research on the utilization of NVM for storage devices or for main memory usage, they performed better as compared to other memories. Storage devices and main memory both of them can use non-volatile memory as they maintain management and integration into the system. Many researches on NVM states that these memories can be used as main memory in the computing system.

Recent research has stated that non-volatile memory (NVM) uses storage and main memory. Because of this integration, the system performance is much enhanced than that of volatile memory used by systems in earlier stages.

If this framework is implemented in recent systems, then we can increase the performance of computer system much higher and with lower cost.

B. Phase Change Memory

Researchers have predicted that Phase Change Memory (PCM) will be used as main memory in future systems. It is expected that this change can benefit in power consumption and superior performance and swift progress in density of phase change memory if we replace the hard disk or Dynamic random access memory with phase change memory. PCM is simply high speed non-volatile storage technology. Phase change memory and Spin-transfer torque random access memory both are widely used non-volatile memory.

Memories in the past were fast, byte addressable and volatile – lost data on system reboots.

But Future memories are faster, byte addressable and are persistent – it means data doesn’t lose on system reboots.

C. Journaling file System

Journaling file systems (JFS) are generally used in new computer systems as they provide higher consistency at reasonable cost. They are especially designed for PCM devices.

Introducing a new file system to phase change memory called Shortcut-JFS. It is very helpful to lesser the write traffic to PCM. The aim of this journaling file system is to slighter the extra writes and boost the I/O performance than present journaling schemes without any failure of consistency.

VII. Conclusion

As far, we have discussed in this article about in-memory file system with SIMFS architecture. We projected a File Virtual Address Space framework.

We came to know that it has long lasting impact on future design of any in-memory file system. The throughput of SIMFS framework approaches the bandwidth of the memory bus. In comparison with all the existing in-memory file system, SIMFS outcomes were enhanced and better than other file systems.

Based on this framework, we are now working on many applications including, user-space file systems, hybrid file systems, new swapping mechanism, distributed in-memory file system, in memory database, etc.

References

[1] M. Jung, J. Shalf, and M. Kandemir, “Design of a large-scale storage-class RRAM system,” in Proc. Int. Conf. Supercomput., 2013, pp. 103–114.

[2] C. Xu, P.-Y. Chen, D. Niu, Y. Zheng, S. Yu, and Y. Xie, “Architecting 3d vertical resistive memory for next-generation storage systems,” in Proc. IEEE/ACM Int. Conf. Comput.-Aided Des., 2014, pp. 55–62.

[3] S. Longerbeam, M. Locke, and K. Morgan. (2013). Protected ram filesystem [Online]. Available: http://pramfs.sourceforge.net/

[4] S. R. Dulloor, S. Kumar, A. Keshavamurthy, P. Lantz, D. Reddy, R. Sankaran, and J. Jackson, “System software for persistent memory,” in Proc. 9th ACM Euro. Conf. Comput. Syst., 2014, pp. 1–15.

[5] X. Wu and A. L. N. Reddy, “Scmfs: A file system for storage class memory,” in Proc. Int. Conf. Supercomput., 2011, pp. 1–11.

[6] H. Kim, J. Ahn, S. Ryu, J. Choi, and H. Han, “In-memory file system for non-volatile memory,” in Proc. Res. Adaptive Convergent Syst., 2013, pp. 479–484.

[7] S. Oikawa, “Integrating memory management with a file system on a non-volatile main memory system,” in Proc. 28th Annu. ACM Symp. Appl. Comput., 2013, pp. 1589–1594.

[8] E. Lee, S. H. Yoo, and H. Bahn, “Design and implementation of a journaling file system for phase-change memory,” IEEE Trans. Comput., vol. 64, no. 5, pp. 1349–1360, May 2015.

[9] Eunji Lee. “Is Buffer Cache Still Effective for High Speed PCM (Phase Change Memory) Storage?”, 2011 IEEE 17th International Conference on Parallel and Distributed Systems, 12/2011

[10] Edwin H.-M. Sha, Yang Jia, Xianzhang Chen, Qingfeng Zhuge, Weiwen Jiang, Jiejie Qin. “The design and implementation of an efficient user-space in-memory file system”, 2016 5th Non-Volatile Memory Systems and Applications Symposium (NVMSA), 2016 Publication

[11] Wu, Xiaojian, Sheng Qiu, and A. L. Narasimha Reddy. “SCMFS : A File System for Storage Class Memory and its Extensions”, ACM Transaction.