ISSN 2047-3338

RAID Dependent Performance on Storage & Retrieval of Digital Forensics Meta Data Files with Different File Systems

Bimal Kumar Goja[1] and Prof. Devanad Padha[2]

[1,2]Department of Computer Science and IT, University of Jammu, Jammu, J&K-180006, India

[1]Vimalgoja@yahoo.co.in, [2]dpadha@rediff.com

***Abstract*– Globally rapid technological developments , fast economic growth, effective storage and effective retrieval of digital forensic meta data is emerging very fast field in modern world and use of this modern technology have necessitated the new emerging trends in digital forensic computing systems. Digital forensics has joined the mainstream in 2003 the American Society of Crime Laboratory Directors–Laboratory Accreditation Board (ASCLD–LAB) recognized digital evidence as a full-fledged forensic discipline. Here the operating systems, high capacity storage hard disks, file systems which are generally operating system dependent and complexity RAID technology will meet the new challenges for the effective storage and effective retrieval of digital forensics meta data (DFMD) files and it is a prerequisite of current and future computer systems. In this paper we analyzed the performance of Meta data files, SVM model interface, different operating systems, comprehensive analysis of files across different platforms and other various dimensions of RAID technology for crucial storage.**

***Index Terms*– Digital Forensics Meta Data, Operating Systems Meta Data Files, Storage Virtualization Model, ZFS File System, Btrfs, UFS, MFS, NTFS, FAT and RAID**

I. INTRODUCTION

IN today's world large storing capacity devices, information security is playing a vital role in digital forensic computing [1], [2]. Here storing and retrieval of digital forensics Meta data files with different versions of operating systems under its layered OS architect model gives effective retrieval, storage and high performance to paramount with ever-increasing challenges of modern technology. Some main limitations is that of data storage with different versions of operating systems and storage systems which communicate with outside world through a narrow block-based interface operating system layers [2], [3].File system handling the file allocation table, volumes and the mapping between clusters which are the basic unit of logical storage on a disk at the operating system level [3], [4]. It is also handling the physical location of data in terms of cylinders, tracks and sectors which are the form of addressing used by the drive's hardware controller. The FAT contains an entry for every file stored on the volume that contains the address of the file's starting cluster. Each cluster contains a pointer to the next cluster in the file, or an end-of-file indicator at (0xFFFF), which indicates that this cluster is the end of the file. The first incarnation of FAT was known as FAT12, which supported a maximum partition size of 8MB.Next FAT16, which increased the maximum partition size to 2GB and the bigger capacity hard disk where put in use and are supported by the new wide variety of operating systems emerged, including Windows 95/98/Mac, OS/2, Linux, NTFS, UNIX and some other versions of UNIX.FAT32 shares all of the other limitations of FAT16.FAT32 incorporated for dual-boot environments, although while other operating systems such as Windows NT can't directly read a FAT32 partition, they can read it across the network. It's no problem, therefore, to share information stored on a FAT32 partition with other computers on a network that are running older versions of Windows.

The proposed model support is extended to include the Windows, NTFS, UFS, VFS, Ext2 / Ext3, MFS, Btrfs and ZFS very effectively to Forensic Meta data (FMD) files for effective storage and retrieval. FAT file system and other file systems are completely having different file systems. Volume is a logical division of data, comprising of a number of files. A single hard disk can have multiple volumes and unlike partitions where volumes can span multiple disks. In our study and experiments It was found that NTFS, Btrfs and ZFS is probably the most advanced file system available for computers featuring superior performance, excellent storage, effective retrieval, security, crash protection and the ability to handle large volumes of data .The performance of a hard disk is very important parameter as to storage and the overall speed of the system, a slow hard disk having the potential to hinder a fast processors [5], [6].

Here proposed OS model components are effective in handling the hard disk and determining a number of factors. Chief among them is the RAID hard disk technology and rotational speed of the platters. Disk RPM is also a critical component of hard drive performance because it directly impacts the latency and the disk transfer rate. The faster the

disk spins, the more data passes under the magnetic heads that read the data and the slower the RPM, the higher the mechanical latencies. Hard drives only spin at one constant speed, and for some time most fast EIDE hard disks span at 5,400 rpm, while a fast SCSI drive are capable of 7,200 rpm. Seagate pushed spin speed to a staggering 10,033 rpm with the launch of its Ultra SCSI drive and also the first manufacturer to release an EIDE hard disk with a spin rate of 7,200 rpm. Mechanical latencies, measured in milliseconds, include both seek time and rotational latency. "Seek Time" is measured defines the amount of time it takes a hard drive's read/write head to find the physical location of a piece of data on the disk."Latency" is the average time for the sector being accessed to rotate into position under a head, after a completed seek. It is easily calculated from the spindle speed, being the time for half a rotation. A drive's "average access time" is the interval between the time a request for data is made by the system and the time the data is available from the drive. Access time includes the actual seek time, rotational latency, and command processing overhead time.

The "disk transfer rate" (sometimes called media rate) is the speed at which data is transferred to and from the disk media (actual disk platter) and is a function of the recording frequency. It is generally described in megabytes per second (MBps). Modern hard disks have an increasing range of disk transfer rates from the inner diameter to the outer diameter of the disk under new operating systems layered interface. This is called a "zoned" recording technique. The key of the model is media recording parameters relating to density per platter are Tracks per Inch (TPI) and Bits per Inch (BPI). A track is a circular ring around the disk. TPI is the number of these tracks that can fit in a given area (inch). BPI defines how many bits can be written onto one inch of a track on a disk surface.

The "host transfer rate" is the speed at which the host computer can transfer data across the IDE/EIDE or SCSI interface to the CPU with operating system layered interface. It is more generally referred to as the data transfer rate, or DTR. RPM (revolutions per minute.) average Seek Time and is the average time it takes for the read/write head to move to a specific location. To compute the average seek time, divide the time it takes to complete a large number of random seeks by the number of seeks performed Seek Time.Latency is the time between initiating a request for data and the beginning of the actual data transfer. For example, the average latency of a hard disk drive is easily calculated from the spindle speed, as the time for half a rotation. In communications, network latency is the delay introduced when a packet is momentarily stored, analysed and then forwarded. Access Time is the time interval between the instant that a piece of information is requested from a memory or peripheral device and the instant the information is supplied by the device. Access time includes the actual seek time, rotational latency, and command processing overhead time in MBps (Megabytes per second) a performance measure used for mass storage devices. Hard disk drives which have EIDE drives generally have a higher TPI than SCSI drives. It is further founded that high level storage capacity hard disks and effective retrieval of forensics meta data files (FMD) can be resolved by implementation of RAID technology hard disks.

RAID for Redundant Array of Independent Disks (originally Redundant Array of Inexpensive Disks), is a technology that provides increased storage and reliability through redundancy [7], [8]. This is achieved by combining multiple disk drive components into a logical unit, where data is distributed across the drives in one of several ways called RAID levels and the layered interface OS model "storage virtualization model " is capable to handle the files to store bigger data , recognize and retrieve effectively on RAID disks. Eerily work was first done by David A. Patterson, Garth A. Gibson, and Randy Katz at the University of California, Berkeley in 1987 as Redundant Arrays of Inexpensive Disks. RAID is now used as an umbrella for high computing; bigger capacity and different file format schemes that can be divide and replicate data among multiple physical disk drives under the SVM interface. The RAW format is an image format, which contains data that is minimally processed from the image sensor of camera and scanners. When we click a picture using our camera the image is saved in RAW file format. When we attach the camera to our computer, it is easily accessed by converting this RAW file to another digital forensic Meta data file format which is done by operating system interface. However we need some applications to open RAW file on our computer easily , like windows operating system uses Coral Paint Shop Photo ProX3 and TSK kit for NTFS and Linux. The operating system layered interface uses application which is open source and capable of handling RAW files, AFF, Encase and changing them to jpg, bmp, gif, pdf, tiff and many other file formats [7].

Encase (.E01) storage format for compressed meta data files which is compressed format developed by Expert witness / guidance software used for compressed , split files across multiple volumes (file.E01 , file.E02 , etc.).Easiest format to work with fast and very widely handled by all tools, supporting many file systems (FAT, NTFS and ext2) which can not have files lesser than 4 Gb.

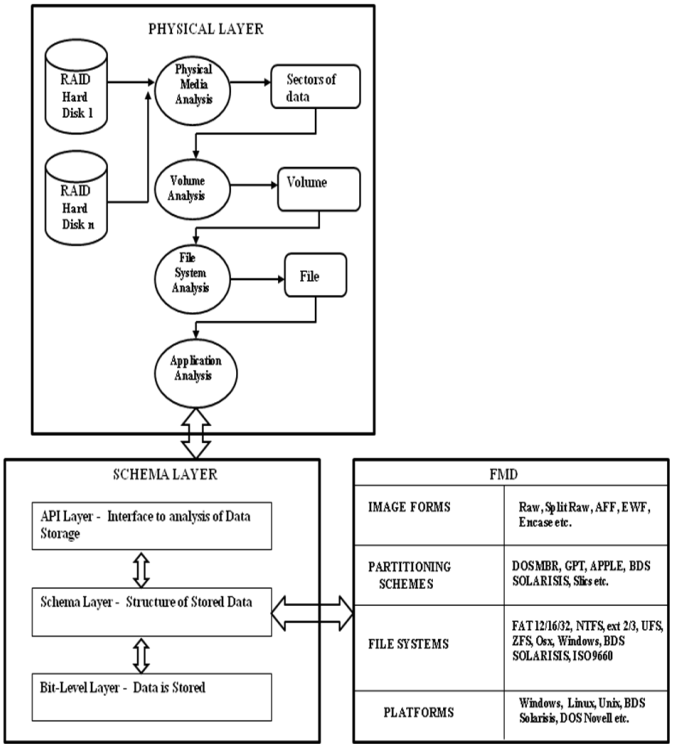
AFF storage format which is compressed open source format which can store images as a single file > 2 GB or as multiple files (.afd format). It further supports encryption and digital signatures, it is extensible also [4], [7]. Here the capital letter words are generic computer words. The DOS, NTFS, Linux, UNIX, Ext2/Ext3 and SUN UNIX operating systems are provided which provide a vast number of filesystems to the user community [8]. In general, one of the major IO challenges faced today is huge storage, scalability, embedded images, high performance ,file journals system, bitmap, scalability, snapshots, compression , RAID support, copy-on-write and many other issues [7], [8]. File systems have to scale in their ability to address these issues and efficiently manage large IO storage subsystems, as well as in their ability to effectively detect, manage, and repair errors and faults in the IO channel. Two of the newer filesystem technologies that aim at addressing today's IO challenges are

SUN's ZFS file system and the Linux filesystem Btrfs.The goal of this paper is 1st, to introduce the design architecture, and features of Btrfs and ZFS,respectively. 2nd, to compare the two filesystems architecture and RAID to elaborate on some of the key performance by design concepts and experiments with files that are embedded in the IO model frameworks. 3rd to conduct an actual file experiment analysis which is comparing the performance behavior of Btrfs and ZFS under varying RAID conditions, utilizing an identical hardware setup for all the file system.

II. EXPERIMENTAL SETUP

The experimental setup is worked out by incorporating OS layered model interface (Storage Virtualization Model) for storing and retrieving meta data files stored across different operating systems and vital RAID Hard Disk technology in use. The experimental evidences are shown in Fig. 1 and through some tables.

Fig. 1. Experimental Setup



A. RAID Experiments

The Meta data files stored across different capacity of RAID hard disk and different operating systems under Layered OS model. While doing this experiment different results were found.

RAID 0 (block-level striping without parity or mirroring) has no (or zero) redundancy. It provides improved performance and additional storage but no fault tolerance. Any disk failure destroys the array, and the likelihood of failure increases with more disks in the array (at a minimum, catastrophic data loss is almost twice as likely compared to single drives without RAID). A single disk failure destroys the entire array because when data is written to a RAID 0 volume, the data is broken into fragments called blocks. The number of blocks is dictated by the stripe size, which is a configuration parameter of the array. The blocks are written to their respective disks simultaneously on the same sector. This allows smaller sections of the entire chunk of data to be read off the drive in parallel, increasing bandwidth. RAID 0 does not implement error checking, so any error is uncorrectable. Storage capacity in a RAID level 0 array with four disks with each capacity of 1000 GB and the total capacity comes out to array 4000 GB. It is calculated as below:

*C = n*d, C = available capacity, n = number of disks, d = disk capacity*

In RAID 1 (mirroring without parity or striping), data is written identically to multiple disks (a "mirrored set"). While any number of disks may be used, many implementations deal with only 2. The array continues to operate as long as at least one drive is functioning. It was found that with a appropriate operating system support there can be increased read performance and only a minimal write performance reduction and this can be avoided by implementing RAID 1 with a separate controller for each disk in order to perform simultaneous reads and writes which is called multiplexing (or duplexing when there are only 2 disks).

Storage capacity in a RAID level 1 array with four disks with each capacity of 1000 GB and the total capacity comes out to array 2000 GB. It is calculated as below:

$$C = n*d/2$$

In RAID 2 (bit-level striping with dedicated Hamming-code parity), all disk spindle rotation is synchronized, and data is striped such that each sequential bit is on a different disk. Hamming-code parity is calculated across corresponding bits on disks and stored on at least one parity disk.

Storage capacity in a RAID level 2 array with four disks with each capacity of 1000 GB and the total capacity comes out to array 2000 GB. It is calculated as below:

$$C = n*d/2$$

In RAID 3 (byte-level striping with dedicated parity), all disk spindle rotation is synchronized, and data is striped so each sequential byte is on a different disk. Parity is calculated across corresponding bytes on disks and stored on a dedicated parity disk.

Storage capacity in a RAID level 3 array with four disks with each capacity of 1000 GB and the total capacity comes out to array 3000 GB. It is calculated as below:

$$C = (n-1)*d$$

RAID 4 (block-level striping with dedicated parity) is identical to RAID 5, but confines all parity data to a single disk, which can create a performance bottleneck. In this setup, files can be distributed between multiple disks. Each

disk operates independently which allows I/O requests to be performed in parallel, though data transfer speeds can suffer due to the type of parity. The error detection is achieved through dedicated parity and is stored in a separate, single disk unit.

RAID 5 (block-level striping with distributed parity) distributes parity along with the data and requires all drives but one to be present to operate; the array is not destroyed by a single drive failure. Upon drive failure, any subsequent reads can be calculated from the distributed parity such that the drive failure is masked from the end user. However, a single drive failure results in reduced performance of the entire array until the failed drive has been replaced and the associated data rebuilt.

Storage capacity in a RAID level 5 array with four disks with each capacity of 1000 GB and the total capacity comes out to array 3000 GB. It is calculated as below:

$$C = (n-1)*d$$

RAID 6 (block-level striping with double distributed parity) provides fault tolerance of two drive failures; the array continues to operate with up to two failed drives. This makes larger RAID groups more practical, especially for high-availability systems. This becomes increasingly important as large-capacity drives lengthen the time needed to recover from the failure of a single drive. Single-parity RAID levels are as vulnerable to data loss as a RAID 0 array until the failed drive is replaced and its data rebuilt; the larger the drive, the longer the rebuild takes. Double parity gives time to rebuild the array without the data being at risk if a single additional drive fails before the rebuild is complete.

Storage capacity in a RAID level 6 array with five disks with each capacity of 1000 GB and the total capacity comes out to array 3000 GB. It is calculated as below:

$$C = (n-2)*d$$

Table 1: Experimental Setup

RAID Level	Min. no. of disks	Size of each drive (GB)	Total available space (GB)
RAID 0	1 2	150 150	150 300
RAID 1	2	150	150
RAID 10 or 0+1	4	150	300
RAID 5	3	150	300
RAID 6	4	150	450
RAID 50	6	150	600
RAID 60	8	150	600

B. File experiment results

In this phase of experiment we stored the files with different Filesystem Technologies and it is found that each file system has its own behavior in handling the files. File system handling the File Allocation Table and the mapping between clusters which are the basic unit of logical storage on a disk at the operating system level and the physical location of data in terms of cylinders, tracks and sectors which are the form of addressing used by the drive's hardware controller.It was found that NTFS , Btrfs and ZFS is probably the most advanced file system available for computers featuring superior performance, excellent storage, effective retrieval, security, crash protection, ability to handle large volumes of data and some other vital features of file handling (Table 2).

Table 2: Experimental Setup

FILE System	Max. file name length	Allowable characters	Max. file size	Max. volume size	File Security
DOS	8 bytes	A-Z, 0-9,	2 GiB	1 GiB At least	No
FAT12	Code units with length [9] 255	Except null with [9], [10] length any Unicode	32 MiB	32 MiB	No
FAT16	Code units with length [9] 255	Except null with [9], [10] length any Unicode	2 GiB	2 GiB or 4 GiB	No
FAT32	Code units with length [9] 255	Except null with [9], [10] length any ASCII	4 GiB	8 TiB	No
MFS	255 bytes	Any byte except null , :	226 MiB	226 MiB	Yes
HFS	31 bytes	Any byte except null , :	2 GiB	2 TiB	Yes
HPFS	255 bytes	Any byte except null , :	2 GiB	2 TiB	Yes
NTFS	255 characters	Any Unicode except null, \/,:,?,*,<,> , “,|,`	16 EiB	16 EiB	No
UFS	255 bytes	Any byte except null	226 TiB	226 TiB	Yes
Ext 2	255 bytes	Any byte except null , /	2 TiB	32 TiB	Yes
ZFS	255 bytes	Any Unicode except null , /	16 GiB	16 GiB	Yes

C. ZFS Results

While handling files with ZFS we found the most important components which are the data integrity and security.It is imperative that information on disk do not suffer from bit rot, silent corruption, or even malicious or accidental tampering. If a power failure occurs during a write operation, the data is not corrupted, and the file system does not resulted in any loss of the block pointers. To accelerate the recovery process, file system incorporate automatically a journaling or logging feature and volume management solution. In case some corruption occurs the data is recovered from the mirror disk and were the the power loss also occurs data is automatically recovered. ZFS also tackles copy-onwrite modifications and constantly checksumming every in-use block in the file system.

1) Copy-on-write transactional model

The ZFS design and represents a combination of a file system and a volume manager together. The file system commands require no concept of the underlying physical disks (because of the storage pool virtualization). All of the high-level interactions occur through the data management unit (DMU), a component of a layered model and eliminates

the need for journaling or logging, as well as for an fsck or mirror resync when a machine reboots unexpectedly.To summarize, ZFS uses a copy-onwrite, transactional object model. All block pointers within the file system contain a 256-bit checksum of the target block, which is verified when the block is read. Blocks containing active data are never overwritten in place; instead, a new block is allocated, modified data is written to it, and then any metadata blocks referencing it are similarly read, reallocated, and written. To reduce the overhead of this process, multiple updates are grouped into transaction groups, and an intent log is used when synchronous write semantics are required.

2) End-to-End Checksumming

To avoid accidental data corruption, ZFS provides memory-based end-to-end checksumming. Most checksumming file systems only protect against bit rot, as they use self-consistent blocks where the checksum is stored with the block itself. In this case, no external checking is done to verify validity.

3) ZFS Scalability

It is data security and integrity paramount, the ZFS file system performs well and increases the limits imposed by modern file systems by using a 128-bit architecture, and by making all metadata dynamic.

4) The 128-Bit Architecture

It is found that the disk drive capacity roughly doubles every nine months and trends continues to require a file systems will require 64-bit addressability in coming years. Instead of focusing on 64-bits, the ZFS designers implemented a 128-bit file system. It is found that the ZFS design provides more capacity than the current 64-bit file systems.

5) Dynamic Metadata

In addition to being a 128-bit based solution, the ZFS metadata is 100 percent dynamic. Hence, the creation of new storage pools and file systems is extremely efficient. Only 1 to 2 percent of the write operations to disk are metadata related, which results in large (initial) overhead savings. There are no static inodes and it is found that data uses the storage pools.

6) RAID-Z

ZFS implements RAID-Z, a solution that is similar to RAID-5, but uses a variable stripe width to overcome the RAID-5 write hole issue.With RAID-5, write operations are performed to 2 or more independent devices, and the parity block is written as a component of each stripe.Since these write operations are non-atomic, a power failure between the data and parity transactions results in the possibility of data corruption. This issue is address with use of parity region logging, a concept that is similar to via battery-backed NVRAM solutions.With the NVRAM solution, the data is written into the NVRAM, and afterwards the data and parity writes are made to disk. Finally, the contents of the NVRAM are released. NVRAM is expensive, and can sometimes turn into the IO bottleneck component. RAID-Z reflects a data/parity scheme that utilizes a dynamic stripe width based approach. Hence, every block reflects a RAID-Z stripe, regardless of the blocksize. This implies that every RAID-Z write and represents a full-stripe write operation.

7) Volumes verses Pooled Storage

Disks do not provide the expected checksum, ZFS reads the parity, and processes the necessary combinatorial reconstruction to determine which disk returned the bad data. It further repairs the damaged disk, and returns good data to the application.

8) Storage Pools

Unlike a traditional UNIX file system that either resides on a single device or uses multiple devices and a volume manager, ZFS is implemented on top of virtual storage pools, labeled the zpools.A pool is constructed from virtual devices (vdevs), each of which is either a raw device, a mirror (RAID-1), or a RAID-(10|Z) group.The storage capacity of all the vdevs are available to all of the file systems in the zpool.

9) ZFS Snapshots

The ZFS copy-on-write model has another powerful advantage; when ZFS writes new data, instead of releasing the blocks containing the old data, it retains them, creating a snapshot version of the file system. ZFS snapshots are created quickly, as all the data comprising the snapshot is already stored. This approach is also space efficient, as any unchanged data is shared among the file system and its snapshots. Writable snapshots (clones) can also be created, resulting in 2 independent file systems that share a set of blocks. As changes are made to any of the clone file systems, new data blocks are created to reflect those changes, but any unchanged blocks continue to be shared, no matter how many clones exist.

IV. BTRFS RESULTS

The Btrfs was designed and implemented with simple components which are modified and optimized btree structures as the main building blocks.

A. Dynamic inode allocation

It is found that dynamic inode allocation is given to storage files by the filesystem.It is based on the actual workload where additional inodes are created and allocated. Btrfs inodes are stored in struct btrfs_inode_item.The Btrfs inode structure is relatively small, and is not containing any embedded file data or extended attribute data.

B. Btrfs Snapshots & Subvolumes

Btrfs subvolumes reflect a btree that stores files and directories .Subvolumes can be given a quota of blocks, and

once this quota is reached, no new write operations are allowed. All of the blocks and file extents inside of the subvolumes are reference counted to allow snapshot operations. Experiment done which shows up to 264 subvolumes can be created per Btrfs filesystem.In Btrfs, snapshots are identical to subvolumes, but their root block is initially shared with another subvolume. When the snapshot is taken, the reference count on the root block is increased, and the copy-on-write Btrfs transaction mechanism ensures changes made in either the snapshot or the source subvolume are private to that root. Snapshots are writable active updates to the snapshot are possible, and they can be snapshotted again (any number of times). The Btrfs snapshots are basically utilized either as backups or as fast emergency copies of the existing data set. Btrfs provides vast flexibility in regards to snapshot functionalities.

C. Btrfs Copy-On-Write

Data, as well as metadata in Btrfs are protected with copy-on-write logging. Once the transaction that allocated the space on disk has committed, any new write operations to that logical address in the file or btree will go to a newly allocated block, and block pointers in the btrees and superblocks will be updated to reflect the new location. Btrfs also utilizes the copy-onwrite mechanism in conjunction with snapshot updates.

D. Btrfs RAID Support

Btrfs supports RAID-0, RAID-1, and RAID-10, respectively. Btrfs allows adding devices (physical disks) to the file system after the original filesystem has been created (the dynamic inode allocation feature is paramount to support this). Further, Btrfs allows for dynamically removing devices from an existing, mounted filesystem. Btrfs further provides file system check and defragmentation options that can be invoked while the filesystem is in use. From a metadata perspective we found that the following options are supported right now.

- RAID-0 - the metadata is appended across all devices.
- RAID-1 - the metadata is mirrored across all devices.
- RAID-10 the metadata is appended and mirrored across all devices.
- Single – the metadata is placed on a single device.

E. Compression

Btrfs provides compression mechanisms focusing on saving disk space, and potentially improving IO performance (the zlib kernel capabilities are being used. Btrfs has significant impact on actual IO performance and has no data copy on write. There is no data checksums and compress which has to turn on.

V. BTRFS VERSES ZFS PERFORMANCE

ZFS as well as Btrfs reveal some design and implementation components that ultimately govern their respective IO performance. In general, a volume manager represents a layer of software that groups a set of block devices for protection and device aggregation purposes. A filesystem represents an abstraction layer that manages such a block device by utilizing a portion of physical memory and layered model interface.From a user space perspective, applications issue read and writes requests to the filesystem, and the filesystem generates IO operations to the scenario of block devices. ZFS collapses these 2 functions into a single entity. ZFS manages a set of block devices (the leaf vdev components), normally groups them into so-called protected devices (such as RAID-Z), and aggregates the top-level vdevs into pool components. The top-level vdevs can be added to a pool at any time.

Objects that are stored in a pool are dynamically striped onto the available vdevs. Associated with the pools, ZFS manages a number of lightweight filesystem objects. A ZFS filesystem can basically be described as a set of properties that are associated with a given mount point.The properties of a ZFS filesystem include the quota (maximum size) and reservation (guaranteed size) as well as attributes such as compression . ZFS files smaller than the recordsize are stored by utilizing a single filesystem block (FSB). The FSB is of variable length but is determined in multiple of a disk sector. Larger files are stored utilizing multiple FSB's, each FSB of recordsize bytes.

Currently, the default FSB size equals to 128KB.In other words, the FSB reflects the basic unit as managed by ZFS (the checksum is applied to the FSB). This gives ZFS high performance.The Btrfs filesystem is still undergoing major development changes. As a matter of fact, the actual disk format has not yet been finalized. In a nutshell, the Btrfs filesystem provides and supports writable snapshots, subvolumes, object-level mirroring and, data checksums, compression, online filesystem checks and defragmentation features.

It is further also found that Solid-state drive (SSD) technologies becoming increasingly common, there is also an SSD optimized mode that aims at increasing the performance potential of Btrfs but we found that Sun's ZFS file system is most capable of handling this feature and found most innovative traditional file system. ZFS file system is also been found best product currently in the market also. Network Appliances' snapshots and object-based storage management, transactions, and checksumming features influenced ZFS.

VI. DISCUSSIONS

The test can be conducted further which will provide an overview of the most other important parameters of standard RAID levels.

Array space efficiency is given as an expression in terms of the number of drives n this expression designates a value between 0 and 1, representing the fraction of the sum of the drives capacities that is available for use.

VII. CONCLUDING REMARKS

Big data storage, effective retrieval and the distribution of data across multiple drives is managed by hardware and operating system interface and main solution is a part of SVM model. ZFS file system and RAID implementations are very effectively provided by model interface operating system which is found capable of storing and effective retrieval of forensics Meta data files. It is also found that ZFS supports equivalents of RAID 0, RAID 1, RAID 5 (RAID Z), RAID 6 (RAID Z2), and a triple parity version RAID Z3, and any nested combination of those like 1+0. ZFS is the native file system on Solaris, and also available on FreeBSD. It is also found that a software RAID controller that is built into an operating system usually uses proprietary data formats and RAID levels. Most software implementations allow a RAID to be created from partitions rather than entire physical drives.

VIII. FUTURE SCOPE

The future scope of this study will be incorporated in effective and big investigation evidences and the common forensic analysis, in which evidence is recovered to support or refute a hypothesis before a criminal court.

This study will also play a vital role for intuitions and organizations in investigating computer security incidents, troubleshooting and some information technology (IT) operational problems by providing practical guidance on performing high computer computations, bigger storage resources, including files, operating systems (OS), network traffic, and applications.

IX. CONCLUSIONS

i). ZFS file system found very effective and efficient across all levels of RAID hard disk technology.
ii). The files across RAID disk technology provides the Failure-resistant (systems that protect against loss of data due to disk failure).
iii). Failure-tolerant (systems that protect against loss of data access due to failure of any single component).
iv). Disaster-tolerant (systems that consist of two or more independent zones, either of which provides access to stored data).
v). Reconstruction of failed drive content to a replacement drive
vi). Protection against data loss due to a "write hole".
vii). Disk automatic swap and hot swap.
viii). Protection against data loss due to cache failure.
ix). Protection against data loss due to external power failure.
x). Faster access during read / write operation.

REFERENCES

[1] Brian Carrier," Digital investigation foundation", File System Forensic Analysis, Publisher: Addision Wesley professional, pub dates: 17th arch.2005, ch. 1, pp12-21.
[2] Brian Carrier, "File System Forensic Analysis", Science direct, Journal of digital forensic, 17th Agust.2005, volume1, issue No.2, pp9-4.
[3] Nelson et al., "Meta data file System", International journal of Digital evidence, 11TH March 2004, volume 1, issue No. 2, pp17-29.
[4] Nelson et al., "Meta data file System", International journal of Digital evidence, 19th September.2006, volume 2, issue No.3, pp23-47.
[5] S. Ross, "Archival digital preservation and methodological Preservation digital libraries", 11th European conference, Seamus Ross, HATII University of Glasgow, 17th September. 2007.
[6] Van der Hoeven , Jeffrey, Bram Lohman, and Verdegem Remco , "Emulation for Digital Preservation in Practice" , The Results in International Journal of Digital Curation , 7th September.2007, volume 2, issue No.2 , pp12-23.
[7] Matthew G, Kirschenbaum, Richard Ovenden, Gabriela Redwine, "Presevation of strategy of digital Forensic meta data", 5th Australian Digital forensic conference, Edith Cowan university, Mount lawlay campus, 3rd December.2007.
[8] Mason, C., "The Btrfs Filesystem ", The Orcale cooperation, 2007.

Bimal Kumar Goja: Technical Officer Department of Computer Science & IT, University of Jammu, Jammu J&K, India.