Comparative study of File systems(NTFS, FAT, FAT32, EXT2, EXT3, EXT4)

Vaibhav(19BCE0723)
B.Tech Computer Science And Engineering
Vellore Institute Of Technology,Vellore
Tamil Nadu,India
vaibhav.2019a@vitstudent.ac.in

Abstract: Over the years, hard drives and the systems used to store data on them have constantly evolved. There are Windows file systems and Linux file system. And have several advantages and disadvantages.File systems have traditionally been a major area of research and development. This is evident from the existence of over 50 file systems of varying popularity in the current version of the Linux kernel. Windows 2000 supports several file systems, the most important of which are FAT-16, FAT-32, and NTFS (NT File System)

Keywords-NTFS,FAT,FAT32,EXT2.

I. INTRODUCTION:

In today's world everything revolves around data. Data is critical for day-to-day operation of any system. Data management is taken care of by file systems which reliably store data on disks[1]. Microsoft Windows OS use two major file systems: FAT, inherited from old DOS with its later extension FAT32, and widely-used NTFS file systems. Recently released ReFS file system was developed by Microsoft as a new generation file system for Windows 8 Servers.

It serves as a database for all kinds of data like structured, unstructured and semi structured. It helps developers to transfer data in many different ways. It is developed by Microsoft to provide a link between windows vista's application layer and NTFS-New Technology File System. It does not serve as alternative to NTFS of Microsoft's.

II Window file systems

Microsoft Windows had 2 widely used file systems. They are

1) NTFS- New Technology File System:- This is the present file system which is given as default to the Windows Operating Systems.
2) FAT-File Allocation Table :- This file system is inherited and developed from old DOS and has been extended to several later versions like exFAT which is also called extended File systems.

A. FAT:

FAT-File allocation Table was introduced by Microsoft in 1980's. At that time it was one of the simplest file systems types and versions. It consists of (i)File System Descriptor Sector(boot sector), (ii)File System Block Allocation Table and (iii)Plane storage space for storing data in files, folders etc. FAT use directories to store the files and folders. These arrays are of 32-byte size which define a file or any external attribute of that file. It also makes a record of first block of the file. The next blocks are found easily through the Block Allocation Table by using it as a linked list. The Block Allocation Table contain an array of block descriptors. Zero value indicates that the block is not used and a non-zero one relates to the next block of a file or a special value for the file end.[2]

B. Version of FAT file system:

1) FAT12-FAT12 is the file system that my 32 MB SD card was formatted with (all SD cards > 4 GB are formatted with FAT32).FAT12 used in different physical formats, but a typical floppy disk at the time was 5.25-inch (130mm), single-sided, 40 tracks, with 8 sectors per track.

2) FAT16-Introduced in 1988, primary file system for MS4.0 up toWindows95 .Support drives size up to 2GB.FAT16 not provide large partition than FAT12. FAT16 used in smaller clusters, making disk usage more efficient, particularly for large numbers of files only a few hundred bytes in size.

3) FAT32 –FAT32 is the latest version of the FAT (file-allocation table) file system.FAT32 was developed by Microsoft to accommodate the ever-increasing size of hard disk drives. It was first introduced with Windows 95 OSR2, and later with Windows 98, and Windows 2000.FAT-32 uses 32-bit block number which supports4GB of block numbers and disks up to 2TB in size.[1]

The numbers 12, 16 and 32 in FAT12, FAT16, FAT32 represent the number of bits used to enumerate a file system block. This means that FAT12 can use up to 2^12=4096 different block references, where as FAT16 and FAT32 can use up to 2^16=65536 and 2^32=4294967296 respectively. The actual maximum count of blocks is actually less and depends file system driver.

FAT12 and FAT16 are used in old floppy disks and they are been out-dated now.
FAT32 is still widely used in memory cards and USB's. FAT32 is supported in smartphones, digital camera and other portable devices.

FAT32 can be used on Windows compatible external storages or disk partitions with the size less than or equal to <= 32 GB. Windows cannot create a FAT32 file system which would be larger than 32 GB, whereas Linux supports the size up to 2 TB and which further doesn't allow to create files the size of which exceeds 4 GB. To overcome this problem, exFAT was introduced by Microsoft to overcome limitations in sizing of files or partitions.[2]

C. Differences in FAT16 and FAT32:

1. The foremost difference among FAT16 and FAT32 is the logical partition size.[3] FAT32 do not have the limitation of the 2-GB logical drive because it can extend a single logical drive with a capacity of at least 127 GB.

2. The root directory folder on a FAT32 drive is a normal cluster chain, so it can be found any where on the volume. For this reason, FAT32 does not constrain the number of entries in the root folder.

3. FAT32 uses small clusters (4 KB for drives up to 8 GB), causing 10 to 15 percent additional efficient use of disk space compared to large FAT16 drives. FAT32 also decreases the resources essential for the computer to activate.[3]

D. NTFS

NTFS-New Technology File System came into existence in 1993 with Windows NT OS and is currently the most common file system available for end user computers on Windows.Almost every windows server use this format only.Files in NTFS are stored as a file descriptor in Master File Table. All the details containing entries of file size, name, allocation, etc are stored in this Master File Table. First 16 entries of the master file table are retained for bitmap. This file table also keeps a record of all clusters which are free and used. It also detects the bad clusters called Badclus. The first sector and last sector of the file system contains system settings. This fie system uses 48 bit and 64 bit values as reference files, which makes them supportable to store large data and have high capacity.[4]

E. UDF

UDF(Universal Disk Format) - file system is the industrial based file format for storing information on the DVD (Digital Versatile Disc or Digital Video Disc) optical media.The UDF file system is provided as dynamically loadable 32–bit and 64–bit modules, with system administration utilities for creating, mounting, and checking the file system on both SPARC and IA platforms. The Solaris UDF file system works with supported ATAPI and SCSI DVD drives, CD-ROM devices, and disk and diskette drives. In addition, the Solaris UDF file system is fully compliant with the UDF 1.50 specification.[5]

F. ReFS:

ReFS - Resilient File System is the latest development of Microsoft introduced with Windows 8 OS and now available for Windows 10 OS. This file system architecture absolutely differs from other Windows file systems and is mainly organized in a form of the B+-tree. ReFS has high tolerance to failures due to new features included into the system.And, namely, Copy-on-Write (CoW): no metadata is modified without being copied; data is not written over the existing data, but into new disk space. With any file modifications, a new copy of metadata is stored into free storage space, and then the system creates a link from older metadata to the newer one. Thus, the system stores significant quantity of older backups in different places providing easy file recovery unless this storage space is overwritten.[6]

NOTE :(COMPARISON OF WINDOWS FILE SYSTEM ACCORDING TO THEIR SPEED)

If we need the Windows-only environment, NTFS is the best choice. If we had to exchange files (even occasionally) with a non-Windows system like a Mac or Linux box, then FAT32 will give you fine result and with no data loss, where your file

size should be less than 4GB.

Generally speaking, exFAT drives are faster at writing and reading data than FAT32 drives. All benchmarks show that NTFS is much faster than exFAT. The bottom line is that unless you are 100 percent sure that you will never have a file smaller than 4 GB, format the drive as exFAT.[7]

III. LINUX FILE SYSTEMS

A. Ext

Ext(extended file system) was the first file system in the series of extended file systems.
Extended file system(ext) was implemented in April 1992 and it is the first file system created specially for the Linux kernel. It has metadata structure inspired by the traditional Unix File System (UFS) and was designed and implemented by Remy Card to overcome the disadvantages of the MINIX file system.[8]

B. Ext2

The ext2 or second extended file system is a file system for the Linux kernel. It was initially designed by Remy Card as a replacement for the extended file system (ext).Ext2 was the default filesystem in several Linux distributions, such as Debian and Red Hat Linux, until ext2 is supplanted more recently by ext3, which is completely compatible with ext2 and ext3 is a journaling file system. ext2 is still used as a filesystem for flash-based storage media (such as SD cards and USB flash drives etc.). The Inode is the foundation of all in Ext2 file system, each file and directory has a different inode, which is kept in the inode table. There is an inode bitmap in the data block group; it's used to record the system's allocated and unallocated inodes. Ext2 realize directory with a special structure, which put the file name and the corresponding inode together. This kind of directory structure is stored in the structure of Ext2 _ dir _entry-2; the main fields of this structure as follows:

• inode: inode number

• rec len: the length of directory entry

• name_len: the length of file name

• file_type: file type

• name: file name

The length of the Ext2 Directory Structure is variable. The last name field of the structure is Variable-length array of Ext2 NAME LEN characters, that's why it varies. Rec len field is able to point to the next valid directory entry, because you can get the start address of the next valid directory entry with the start address of the current directory entry and Rec _len field as an offset.[10]

C. Ext3

Third extended file system(ext3) is the commonly used Linux file system. Ext3 is introduced in 2001 which is developed by Stephen Tweedie. Ext3 was available starting from Linux kernel(v2.4.15). Maximum file size of ext3 ranges from 16GB to 2TB. There are three types of journaling available in ext3 file system.The main advantage of ext3 over ext2 is journaling, which improves reliability and eliminates the need to check the file system after an unclean shutdown.

D. Ext4

Ext4 was developed to extend storage limits and to add other performance improvements.It supports the maximum file size as comparative to other file systems.Maximum file size of ext4 ranges from 16 GB to 16 TB. Some new features are added in Ext4 such as it supports multiblock allocation, delayed allocation, journal checksum.These new features have improved the performance and reliability of the file system when compared to ext3.[11] ext4 – is the result of evolution ext3, the most popular file system in Linux. In many ways, Ext4 is a big step forward compared to ext3, ext3 than was in relation to the ext2 . The most significant improve-ment compared to ext3 ext2 was logging, while ext4 involves important changes in data structures, such as for storing data files.One improvement is to increase the maximum size of a single file for the FS. Today, the maximum size ext3 file system is 16 terabytes, and the file size is lim-ited to 2 terabytes. In Ext4 adds 48-bit block address-ing, which means that the maximum size of this file system is equal to one exabyte and files can be up to 16 Terabytes. 1 EB (exabytes) = 1,048,576 TB (tera-bytes), 1 EB = 1024 PB (petabytes), 1 PB = 1024 TB, 1 TB = 1024 GB. 48-bit addressing of blocks used for a number of limitations that need to be removed, to make Ext4 fully 64-bit, and this problem was not put before the Ext4. The data structures in the Ext4 designed taking into account the changes required, so one day in the future, support for 64 bits per Ext4 will appear. In the meantime, we have to settle for one ex-abytes.[16]

E. XFS

XFS is a high-performance 64bit journaling file system created by Silicon Graphics,Inc(SGI) in 1993.It is supported by most Linux distributions,some of them uses XFS as the default file system.SGI's IRIX operating system(starting from v5.3) uses XFS as the default file system.

XFS excels in the execution of parallel input/output (I/O) operations.XFS enables extreme scalability of I/O threads, file system bandwidth, and size of files when spanning multiple physical storage devices.

XFS features include striped allocation of RAID arrays,file system journaling,variable block sizes,direct I/O,guaranteed-rate I/O,snapshots,online defragmentation,online resizing. A uniquefeature to XFS is the pre-allocation of I/O bandwidth at a pre-determined rate,this is suitable for many real-time applications. However, this unique feature was supported only on IRIX operating system and only with specialized hardware.[12]

F. JFS

Journaled File System(JFS) is a 64-bit journaling file system created by IBM. There are versions for AIX, eComStation, OS/2, and Linux operating systems. The latter is available as free software under the terms of the GNU General Public License (GPL).

In the AIX operating system, there are two generations of JFS file system namely JFS (JFS1) and JFS2respectively. The second generation exists in operating systems such as OS/2 and Linux and is simply called as JFS.W**e** should not be confused with JFS in AIX that refers to JFS1.[13]

G. ReiserFS

ReiserFS is a general-purpose journaled file system formerly designed and implemented by a team at Namesys led by Hans Reiser..ReiserFs was introduced in Linux kernel(v2.4.1) ReiserFS is used as the default file system on Elive, Xandros, Linspire, and YOPER Linux distributions. ReiserFS is used as default file system in Novell's SUSE Linux Enterprise until Novell decided to move to ext3 on October 12, 2006 for future releases.

Namesys considered ReiserFS as stable and feature-complete and, with the exception of security updates and critical bug fixes, ceased development on it to concentrate on its successor, Reiser4.[14]

H. Reiser4

Reiser4 is a file system created and developed by Namesys and it is successor to ReiserFS. The creation of Reiser4 was backed by the Linspire project as well as DARPA. What makes Reiser4 special is its multitude of transaction models[15].

IV. Ext4 file system in Linux Environment:Features and Performance Analysis

A. Ext4

Ext4 was developed to extend storage limits and to add other performance improvements.It supports the maximum file size as comparative to other file systems.Maximum file size of ext4 ranges from 16 GB to 16 TB. Some new features are added in Ext4 such as it supports multiblock allocation, delayed allocation, journal checksum.These new features have improved the performance and reliability of the file system when compared to ext3.

<u>New features added to ext4 are</u>

1) 64 bit file system

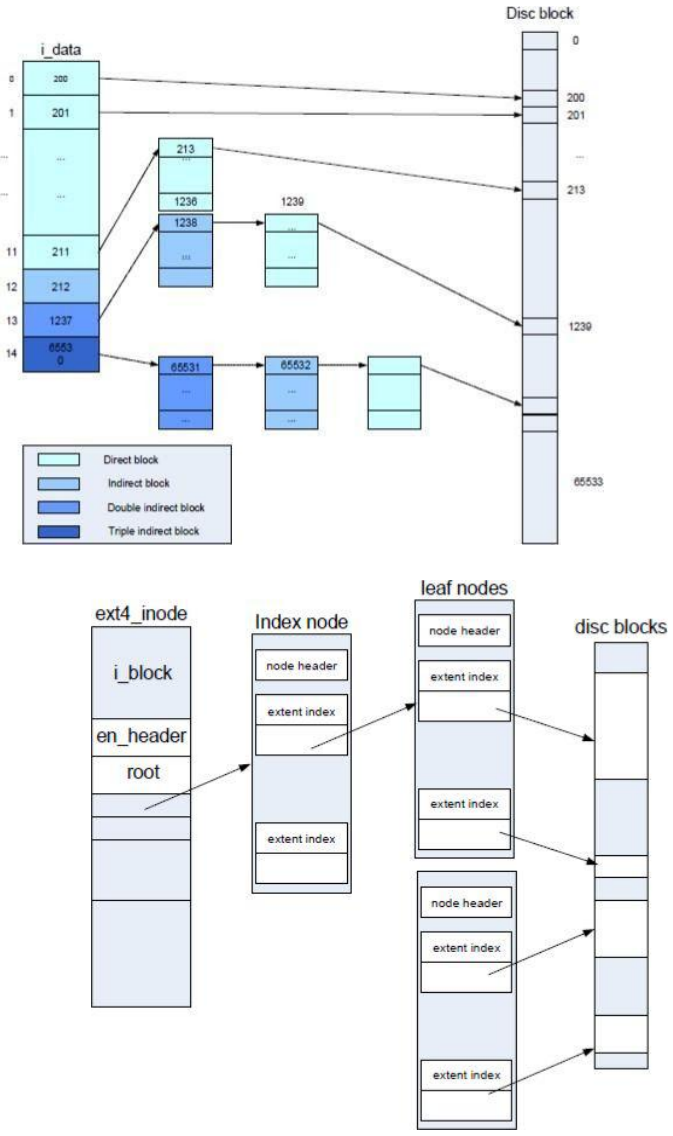
Ext4 is 64bit filesystem allowing file size upto 16TB

2) Inode size of 256 bytes

3)Extents

Major difference between ext3 and ext4 file system is the way that the block numbers are stored with data files.Ext3 uses indirect mapping blocks.ext4 remembers the number of blocks in extent (descriptor that represents a continuous series of physical blocks)

Ext2/ext3mapping



Ext4 mapping

4)Directory Scalability

In Ext3 directory entries are stored in a linked list,which is very inefficient for directories with large number of enries.The directory indexing feature addresses this scalability issue by storing directory entries in Htree data structure,which is specialized BTree like structure using 32bit hashes.Fast lookup time of Htree improves performance on large directories.

5)Block allocation enhancment

Higher fragmentation rates causes greater disk access time affecting overall throughput.It has impact on increased metadata overhead causing less efficient mapping.By improving block allocation techniques we can reduce file system fragmentation.

6) Online defragmentation

There is a possibility that the ext4 file system can still become quite fragmented.Ext4 online defragmentation tool,e4defrag can defragment individual files or entire file system and by this way it helps in avoiding file fragmentation caused with file system aging.

7)Reliabilty

Ext3 is one of the most reliable file systems,therefore developers are putting much effort to make it even more reliable.[9]

"Magnetization (A (m(1)," not just "A/m." Do not label axes with a ratio of quantities and units. For example, write "Temperature (K)," not "Temperature/K."

V. CODE

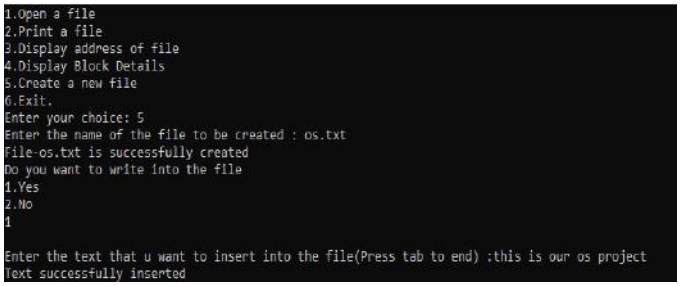
```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#include<conio.h>
#include <string.h>
int s_b[10][2],empty[10];
int mem=500;
char *block[10];
int *address[10];

char name[10][30];
void open()
{
char fname[30],c;
int size=0,i;
printf("Enter .txt file name\n");
scanf("%s",fname);;
FILE *inputf;
inputf = fopen(fname,"r");
if (inputf == NULL)
{
printf("\nFile unable to open ");
exit(0);
}
rewind(inputf);
```

```
while(c!=EOF)
{
c=fgetc(inputf);
size=size+1;
}
printf("The size of given file is : %d\n", size-2);
if(mem>=size)
{
int n=1,parts=0,m=1;
while(address[n]!=0)
n++;
strcpy(name[n],fname);
s_b[n][1]=size;
int bnum=size/50;
if(size%50!=0)
bnum=bnum+1;
s_b[n][2]=bnum;
mem=mem-(bnum*50);
int *bfile=(int*)malloc(bnum*(sizeof(int)));
address[n]=bfile;
printf("Number of blocks required: %d\n",bnum);
rewind(inputf);
c = fgetc(inputf);
while(parts!=bnum && c!=EOF)
{
int k=0;
if(empty[m]==0)
{
char *temp=block[m];
while(k!=50)
{
*temp=c;
c=fgetc(inputf);
temp++;
k=k+1;
}
*(bfile+parts)=m;
parts=parts+1;
empty[m]=1;
}
else
m=m+1;
}
printf("File created\n");
printf("\n");
fclose(inputf);
}
else
printf("Not enough memory\n");
}
int filenum(char fname[30])
{
int i=1,fnum=0;
while(name[i])
{
if(strcmp(name[i], fname) == 0)
{
fnum=i;
break;
}
i++;
}
return fnum;
}
void blocks()
{
int i;
printf(" Block adddress empty/free\n");
for(i=1;i<=10;i++)
printf("%d. %d - %d\n",i,block[i],empty[i]);
printf("\n");
}
void file()
{
int i=1;
printf("File name size address\n");
for(i=1;i<=10;i++)
{
if(address[i]!=0)
printf("%s %d %d\n",name[i],s_b[i][1],address[i]);
}
printf("\n");
}
void print()
{
char fname[30];
int i=1,j,k,fnum=0;
printf("Enter the file name: ");
scanf("%s",fname);
fnum=filenum(fname);
if(fnum!=0&& address[fnum]!=0)
{
int *temp;
temp=address[fnum];
```

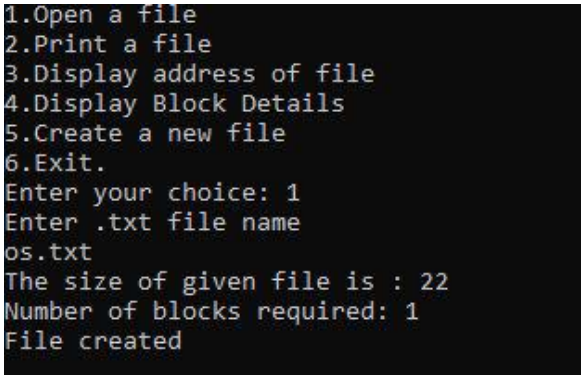
```
printf("Content of the file %s is:\n",name[fnum]);
int b=(s_b[fnum][2]);
for(j=0;j<b;j++)
{
int s=*(temp+j);
char *prt=block[s];
for(k=0;k<50;k++)
{
printf("%c",*prt);
prt++;
}
}
printf("\n");
printf("\n");
}
else
printf("File not available:/n");
}
void create() //to create a file
{
FILE *inputf;
char fname[30],ch;
int a;
printf("Enter the name of the file to be created : ");
scanf("%s",fname);
inputf=fopen(fname,"w");
printf("File-%s is successfully created\n",fname);
printf("Do you want to write into the file\n1.Yes\n2.No\n");
scanf("%d",&a);
printf("\n");
if(a==1)
{
 printf("Enter the text that u want to insert into the file(Press
tab to end) :");
 while((ch=getchar())!='\t')
 {
 fputc(ch,inputf);
 }
 printf("Text successfully inserted\n");
}
fclose(inputf);
}
int main()
{
 char*buffer =(char*)malloc(500); //Memory created-500
bytes
int choice,i;
char *temp;
if (buffer == NULL)
{
fputs ("Memory error",stderr);
exit (2);
}
temp=buffer;
block[1]=buffer;
empty[1]=0;
for(i=2;i<=10;i++)
{
block[i]=block[i-1]+50;
empty[i]=0;
}
while(1)
{
printf("1.Open a file\n");
printf("2.Print a file \n");
printf("3.Display FAT table\n");
printf("4.Display Block Details\n");
printf("5.Create a new file\n");
printf("6.Exit.\n");
printf("Enter your choice: ");
scanf("%d",&choice);
switch(choice)
{
case 1:
open();
break;
case 2:
print();
break;
case 3:
file();
break;
case 4:
blocks();
break;
case 5:
create();
break;
case 6:
exit(1);
}
}
return 0;
}
```

VI. OUTPUT

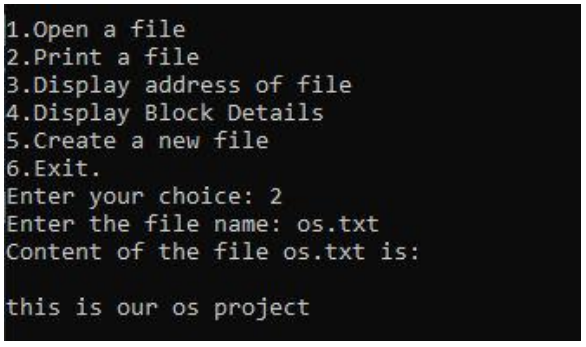
A.TO CREATE A FILE AND INSERT DATA



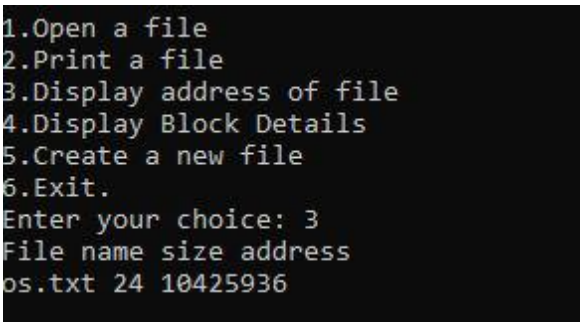
B. TO OPEN AN EXISTING FILE



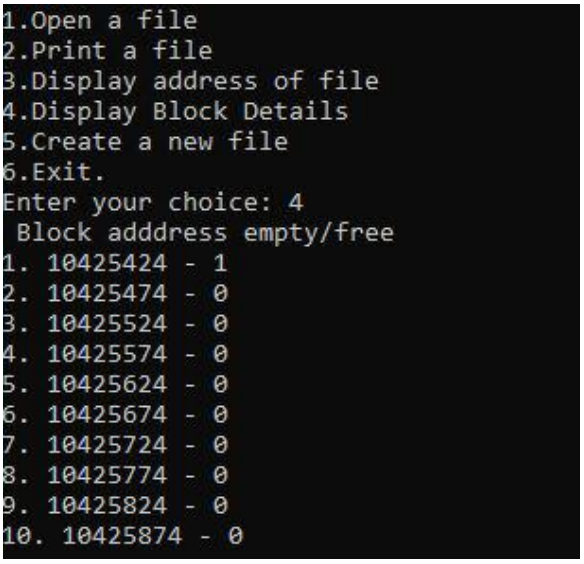
C. TO PRINT THE DATA PRESENT IN FILE



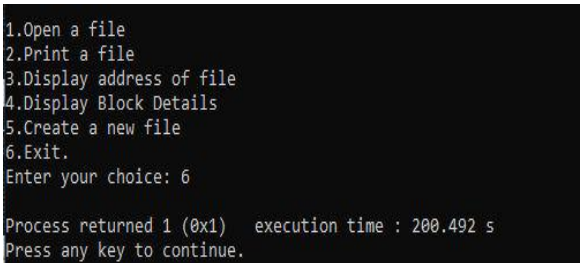
D. TO DISPLAY ADDRESS OF FILE



E. TO DISPLAY BLOCK DETAILS OF FILE



F. TO EXIT



VII. CONCLUSION

If we need the Windows-only environment, NTFS is the best choice. If we had to exchange files (even occasionally) with a non-Windows system like a Mac or Linux box, then FAT32 will give you fine result and with no data loss, where your file size should be less than 4GB.

While file transfer speed and maximum throughput is

limited by the slowest link (usually the hard drive interface to the PC like SATA or a network interface like 3G WWAN), NTFS formatted hard drives have tested faster on benchmark tests than FAT32 formatted drives

Generally speaking, exFAT drives are faster at writing and reading data than FAT32 drives. All benchmarks show that NTFS is much faster than exFAT. The bottom line is that unless you are 100 percent sure that you will never have a file smaller than 4 GB, format the drive as exFAT

REFERENCES:

[1] Akash Bundele and Prof. Dr. S. E. Yedey, "Comparative study of File systems(NTFS, FAT, FAT32, EXT2, EXT3, EXT4)",August 2018(http://www.ijraset.com/fileserve.php?FID=18507).

[2] "File Allocation Table"(https://en.wikipedia.org/wiki/File_Allocation_Table).

[3] Riya Madaan, Rakesh Kumar and Girdhar Gopal, " File Allocation and Recovery in FAT16 and FAT32", December 2016 (https://www.ijser.org/researchpaper/File-Allocation-and-Recovery-in-FAT16-and-FAT32.pdf).

[4] "NTFS" (https://en.wikipedia.org/wiki/NTFS)

[5] "The Universal Disk Format (UDF) File System"(https://docs.oracle.com/cd/E19683-01/817-6960/fsoverview-8/index.html)

[6] "Understanding file systems"(https://www.ufsexplorer.com/articles/file-systems-basics.php)

[7] FAT32 vs. NTFS: Choose Your Own Format(https://in.pcmag.com/storage/72354/fat32-vs-ntfs-choose-your-own-format)

[8]"Extendedfilesystem" (https://en.wikipedia.org/wiki/Extended_file_system)

[9] Borislav Djordjevic and Valentina Timcenko, " Ext4 file system in Linux Environment: Features and Performance Analysis",2012

(https://pdfs.semanticscholar.org/0dbf/4d459f40322c2c70cc76e5a954de3006d4ff.pdf)

[10] CHEN Wei and LIU Chun-mei "The Analysis and Design of Linux File System Based on Computer Forensic",2010(https://ieeexplore.ieee.org/document/5541078)

[11] "ext4"(https://en.wikipedia.org/wiki/Ext4)

[12] "XFS"(https://en.wikipedia.org/wiki/XFS)

[13]"JFS"(https://techterms.com/definition/jfs)

[14]ReiserFS(https://en.wikipedia.org/wiki/ReiserFS)

[15]Reiser4 (https://en.wikipedia.org/wiki/Reiser4)

[16] Andrey Vladimirovich ostroukh and Alexander Gennadievich Salniy, "Research of Performance Linux Kernel file Systems",2016 (https://www.researchgate.net/publication/282619311_Research_of_Performance_Linux_Kernel_File_Systems)

A Report on Multimedia File Systems

Deepesh Suranjandass(19BCE2210)
B.Tech CSE.Core
Vellore Institute of Technology
Vellore, India
deepesh.suranjandass2019@vitstudent.ac.in

***Abstract*— A multimedia file system is a type of file-system that supports real-time sessions as well as normal disk traffic. Whenever a request is made for the real-time session, the file-system guarantees that starvation does not occur as long as the system does not crash and the user process can read or write data.**

Keywords—Multimedia File System, Mobile File System, Disk Scheduling Algorithms, EDF-SCAN

I. Introduction

A multimedia file-system can store and retrieve multimedia files, i.e. files containing sound and/or video, apart from normal data files. Multimedia files can be accessed in two different ways. They can be accessed either through read or write system calls or through a pair of new system calls that allow the user to process the data at a non-stop constant rate. All in all, this means that the file system makes sure that there are always enough data or free buffer space for the playback or recording process. A file is well laid out for multimedia purposes, when the file is divided into a finite number of large bits/chunks of contiguous data and also if the head-travelled distance from the end of one chunk to the beginning of the next is small.

II. Multimedia generic algorithm

A. *Introduction*

For inclusion of multimedia, the generic algorithm as well as the bin-packing algorithm are combined. This generic solution to let hard disk drives with a single main stream to play and record multiple video streams, handles different disk types and different streams with different bit rates.

B. *Difference between generic algorithm and normal algorithms*

A multimedia file system can be differentiated from a traditional file system in a way that in the former optimized strategies are used for caching and receiving data. Real Time multimedia files are accessed sequentially as it favors data placement in seek optimally.

III. Multimedia Disk Scheduling Algorithms:

In multimedia file systems, disk scheduling is used to meet the goal of all the time critical tasks and the encoding scheme gives the ability to the player to change the playback rate of content of multimedia. Therefore, the buffer requests should be kept low and the aperiodic requests should not undergo starvation.

Due to the immaculate requirements of storage space for continuous media, conventional magnetic storage devices are no longer used or viable. Storage devices like Tapes, used in some traditional file systems are inadequate for multimedia file systems because they cannot provide independent accessible streams, and random access is slow and expensive. The below are the different types of disk scheduling algorithms(both classical and real-time)

A. *FCFS*

This is the simplest strategy in which each request is served in first-come-first-serve basis. All incoming requests are placed at the end of the queue. Whatever number that is next in the queue will be the next number served.

B. *SCAN*

This is also known as the elevator algorithm in which the arm moves in one direction and serves all the request in that direction until there are no further request in that direction.

C. *C-SCAN*

The circular SCAN algorithm works in the same way as SCAN except that it always scans in one direction. After serving the last request in the scan direction, the arm return to the start position.

D. *SSTF*

The SSTF, for shortest seek time first, algorithm simply selects the request closest to the current arm position for service.

E. *SSTF*

The SSTF, for shortest seek time first, algorithm simply selects the request closest to the current arm position for service.

F. *EDF*

The EDF Algorithm is an exact opposite of the FCFS algorithm, as in that it gives the highest priority to the process with the nearest deadline time.

G. *P-SCAN*

In P-SCAN, all the processes are divided into various priority levels and the SCAN algorithm is used within each level.

H. FD-SCAN

In FD-SCAN, the track location of each request from every process with the earliest feasible deadline is used to determine the scan direction. A deadline is feasible if we estimate that it can be met. Each time that a scheduling decision is made, the read requests are examined to determine which have feasible deadlines given the current head position.

I. SCAN-EDF

In SCAN-EDF Algorithm, it utilizes SCAN Algorithm to reschedule tasks in a real-time EDF Schedule. This is one of the best real-time disk scheduling algorithms. Since tasks rescheduled in SCAN-EDF should have the same deadline, its efficiency depends on the number of tasks with the same deadlines. If all tasks have different deadlines, the schedule result of SCAN-EDF would be the same as EDF. In SCAN-EDF , rescheduling is only possible within a local group of requests.

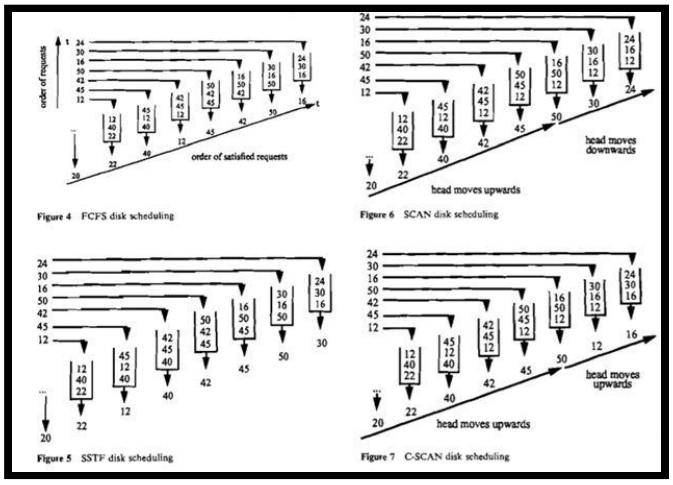


Figure: Different Disk Scheduling Algorithms for Multimedia-File Systems

IV. Case Study on C-Scan and SCAN-EDF

Multimedia file Systems Generally are generally composed of Audio, Video and Animations. These belong to a class of data known as delay sensitive, as they are very sensitive to delay in the presentation of the user. In order to have an acceptable presentation, disk requests deadlines must be met, and a real-time scheduling approach should be used to guarantee the timing requirements for such environment. Two Disk Scheduling Algorithms that meet these standards to a reasonable level are C-SCAN Algorithm and the real-time SCAN-EDF Algorithm.

A. C-SCAN Disk Scheduling Algorithm

This is a more classical approach to solving the Disk Scheduling Problem for Multimedia File Systems. The circular SCAN algorithm works in the same way as SCAN except that it always scans in one direction After serving the last request in the scan direction, the arm return to the start position. These Traditional Disk Scheduling Algorithms do not consider time constraints of I/O Tasks and hence cannot be applied directly to a real-time system. This results in poor performance of classical algorithms in multimedia file system

B. SCAN-EDF Disk Scheduling Algorithm

SCAN-EDF, which utlizes SCAN to reschedule tasks in a real-time EDF schedule, is one of the best real-time disk scheduling algorithms. Assigning priorities to transactions an Earliest Deadline policy minimizes the number of late transactions in systems operating under low or moderate levels of resource and data contention. This is due to the highest priority given to the transactions that have the least remaining time in which to complete. However, the performance of Earliest Deadline steeply degrades in an overloaded system. This is because, under heavy loading, transactions gain high priority only when they are close to their deadlines. Gaining high priority at this late stage may not leave sufficient time for transactions to complete before their deadlines. Since tasks rescheduled in SCAN-EDF should have the same deadline, its efficiency depends on the number of tasks with the same deadlines. If all tasks have different deadlines, the schedule result of SCAN-EDF would be the same as EDF.

Algorithms	Number of Supported Requests		
	minimum	maximum	average
BFI	6	26	15.14
SCAN-EDF	4	24	10.21
EDF	3	24	9.87
Greedy	1	10	4.38

Figure: The maximum number of requests supported by different scheduling policies

C. Program to check the number of movements requried by the head of a disk for C-SCAN and SCAN-EDF

Code:

```
from random import randint

def c_scan(queue, max):
    head = queue[0]
    requests = queue[::]

    print("---- C-SCAN Disk Scheduling Algorithm ----")
    print("Queue is:", queue)
    print("Head is on", head)

    requests.sort()

    # Find the nearest end and cut lists
    if head <= abs(head - max):
        start = ([0] + requests[0:requests.index(head) +
1])[::-1]
        end = (requests[requests.index(head) + 1::] +
[max])[::-1]
    else:
        start = requests[requests.index(head):] + [max]
```

```
        end = [0] + requests[0:requests.index(head):]

    requests = start + end
    res = []
    for i in range(1, len(requests)):
        res.append(abs(requests[i - 1] - requests[i]))

        # Skip difference between start and end
        if (requests[i-1] == max and requests[i] == 0) or
(requests[i-1] == 0 and requests[i] == max):
            res[i - 1] = 0
        print('From {:>3} to {:>3} with {:>3}
seeks'.format(requests[i - 1], requests[i], res[i - 1]))

    return sum(res)


def scan_edf(queue, max):
    # Sort by execution time and then by deadline
    requests = sorted(queue[::], key=lambda x: x[:2])
    head = min(requests, key=lambda a: a[0])[2]

    print("---- EDF-SCAN Real-time Disk Scheduling
Algorithm ----")
    print("Queue:", requests)
    print("Head is on:", head)

    # Filters list by execution time and then by deadline :
    e = []
    temp = []
    last_index = 0
    for i in range(len(requests) - 1):
        temp.append(requests[i])
        # Save sublist while next one isn't different
        if requests[i][0] != requests[i + 1][0]:
            #if the ith element in the queue does not equal
            #the (i+1)th element in the matrix
            #then initialize that value as the last value
            last_index = requests.index(requests[i])
            e.append(temp)
            temp = []

    # Add last part
    # Map list by third value in sublist
    # Copy list to previous name
    e.append(requests[last_index + 1:])
    for i in range(len(e)):
        e[i] = list(map(lambda x: x[2], e[i])) #third value in
the list
    requests = e[::] #make this the array
    # Format by execution time
    print("Formatted list:", requests) #print the formatted
list of only deadline times

    # Set initial direction in which disk's head moves ('r'
or 'l'),
    #this is the unique part to implement SCAN
Algorithm
    dr = 'l' if head <= abs(head - max) else 'r'

    seek = 0
    for req in requests:
        request = [head] + req[::]
        request.sort()

        # Cuts request list by two parts
        if dr == 'r':
            start = request[request.index(head):] + [max]
            end = (request[0:request.index(head):])[::-1]
            dr = 'l'
        elif dr == 'l':
            start = ([0] + request[0:request.index(head) +
1])[::-1]
            end = request[request.index(head) + 1::]
            dr = 'r'

        # Update head position
        request = start + end
        head = request[-1]

        # Count gap between each request in current
deadline
        res = []
        for i in range(1, len(request)):
            res.append(abs(request[i - 1] - request[i])) #this
is just to count the number of seeks for each element
inside requests
            print('From {:>3} to {:>3} with {:>3}
seeks'.format(request[i - 1], request[i], res[i - 1]))

        seek += sum(res) #print the total number of seeks
    return seek


# MAX value of disk head position
# Generate queue with 5 random requests
MAX = 99
queue = [randint(0, MAX) for i in range(20)]
print("Total movements:", c_scan(queue, MAX), '\n')

# Random list with execution time, deadline and position
# Change range to get more or less random values
queue = [[randint(0, 3), randint(1, 10), randint(0, MAX)]
for _ in range(5)]
print("Total movements:", scan_edf(queue, MAX))
```

Output:

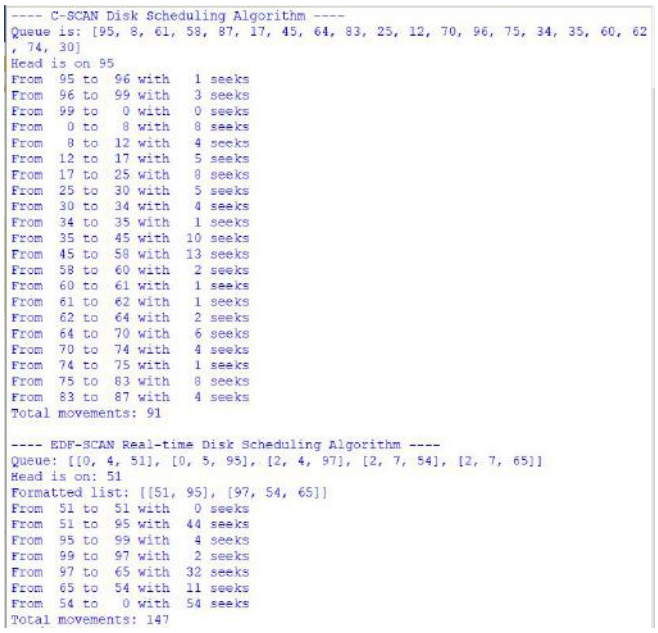


Figure: Output of the code displaying the number of movements for C-SCAN and SCAN-EDF

REFERENCES

[1] Lily B. Mummert, Maria R. Ebling, and M. Satyanarayanan, ‘‘Exploiting Weak Connectivity for Mobile File Access,’’ in Proceedings of the 15th ACM Symposium on Operating System Principles, Copper Mountain (December, 1995).

[2] Peter L. Reiher, John S. Heidemann, David Ratner, Gregory Skinner, and Gerald J. Popek, ‘‘Resolving File Conflicts in the Ficus File System,’’ pp. 183−195 in Proceedings of the Summer USENIX Conference, Boston (June 1994).

[3] B. Callaghan, B. Pawlowski, and P. Staubach, ‘‘NFS Version 3 Protocol Specification,’’ RFC 1813,Network Information Center, SRI International, Menlo Park, CA (June, 1995).

[4] Haiying, S.J., 2010. IRM: Integrated file replication and consistency maintenance in P2P systems. IEEE Transact. Parallel Distributed. Syst., 21(1):100-113.

[5] Aniruddha, B., S. Smaldone and L. Iftode, 2007.FRAC: Implementing role-based access control for network file systems. Proceedings of the Sixth IEEE International Symposium on Network Computing and Applications, (SNCA’ 2007), 12-14 July 2007, Rutgers Univ., Piscataway, pp: 95-104

.

File Allocation Methods

Tanmay Bansal
Vellore institute of technology
India
tanmay.bansal2019@gmail.com

Abstract: Today computer is an integral part of human life. The storage of large amount of data permanently in computer system files is used. In this research paper we discuss the file that is a collection of records or information stored on secondary storage such as hard disk. In computing a file system is used to control how data is stored and retrieved. File system control the files starting and ending locations. The information present in the file can be accessed using access methods. In file any time data is failure with hardware problem for solution file system provide protection with access privileges of users. In files secondary storage space is allocated using file allocation methods. These allocated space in such a manner so that disk space is utilized effectively and files can be accessed quickly. Directory structure is use symbol table of files that stores all the related information about the file it holds with the contents.

Keywords—File System, File Protection, File Access Methods, File Allocation Methods

I. INTRODUCTION

File is a logical collection of information stored on secondary storage such as hard disk. It is a collection of records. Physically, a file is smallest allotment of secondary storage device for example disk. Logically, a file is a sequence of logical records such as a sequence of bits and bytes. Files can be used to contain the data and programs (both source and object programs). Data files can be numeric, alphabetic, alphanumeric or binary. A file has various attributes like name, type, location, size, protection, time and date of creation etc. Computers can store information in several physical forms, depending on which storage device is used. Disks and drums though are the most common devices for this purpose. Since each device has its own characteristics and physical organization, information may be stored in several ways and therefore different views of information are created. To unify all these views of information in the system, a uniform logical view of it was created. This logical view is called a file. It is the job of the OS to map this sequence of words into physical devices. The part of the OS responsible for this is the file system. It is clear that the main objective of the file systems is to free the users of the details of storing the information in the physical devices. That is, when the storage device is changed, from disk to drum for example, the user still sees the same information as before the change. If this is allowed in the system, then we can say that the file system is device dependent.

II. FILE SYSTEM

In computing a file system is used to control how data is stored and retrieved. Without a file system, information placed in a storage area would be one large body of data with no way to tell where one piece of information stops and the next begins. By separating the data into individual pieces, and giving each piece a name, the information is easily separated and identified. Taking its name from the way paper-based information systems are named, each group of data is called a "file". The structure and logic rules used to manage the groups of information and their name is called a "file system". There are many different kinds of file systems. Each one has different structure and logic, properties of speed, flexibility, security, size and more. File systems can be used on many different kinds of storage devices. Each storage device uses a different kind of media. The most common storage device in use today is hard device whose media is a disc that has been coated with a magnetic film. The film has ones and zeros 'written' on it sending electrical pulses to a magnetic "read-write" head. Other media that are used are magnetic tape, optical disc and flash memory. The file system manages access to both the content of files and the metadata about those files. It is responsible for arranging storage space; reliability, efficiency, and tuning with regard to the physical storage medium are important design considerations.

III. ACCESS METHODS OF FILE

Files are used to store data. The information present in the file can be accessed by various methods. Thus, the way of retrieving data from a file is known as access methods. Different systems use different access methods. The various access methods used are:
1. Sequential access
2. Direct access
3. Indexed access

1. Sequential access:- It is the simplest and most commonly used access method. In this information in the file is accessed in the order it is stored in the file one record after the other. The various records are read sequentially one after the other in an order, starting at the beginning to the end of the file. The various records cannot be read randomly out of order we can not skip any record in between. For example reading of 34 record followed by 5 record and then 1 record is not possible in sequential access. A read operation reads the next portion of the file and automatically advances the file pointer. Similarly, a write appends to the end of the file and the file pointer. Similarly, a write appends to the end of the file and the file pointer. Similarly, a write appends to the end of the end of the file and advances to the end of the newly written material (the new end of file). Such a file can be reset to the beginning, and, on some systems, a program may be able to skip forward or backward n records, for some integer n. This scheme is known as sequential access to a file. Sequential access is based on a tape model of a file. Sequential access is convenient when the storage medium is magnetic tape, rather then a disk.
2. Direct Access:- In direct access method it is possible to access the records of a file in any order. For example, if we are reading block 13, we can read block 46 after this and then block

20. Various records are read or write randomly. Direct access is based on a disk model of a file. For direct access, the file is viewed as a numbered sequence of block or records. A direct-access file allows arbitrary blocks to be read or written. Thus, after block 18 has been read, block 57 could be next, and then block 3. There are no restrictions on the order of reading and writing for a direct access file. Direct access files are of great use for intermediate access to large amounts of information. The file operations must be modified to include the block number as a parameter. Thus, we have "read n", where n is the block number, rather than "read next", and "write n", rather that "write next". An alternative approach is to retain "read next" and "write next" and to add an operation; "position file to n" where n is the block number. Then, to effect a "read n", we would issue the commands "position to n" and then "read next". Not all OS support both sequential and direct access for files. Some systems allow only sequential file access; others allow only direct access. Some systems require that a file be defined as sequential or direct when it is created; such a file can be accessed only in a manner consistent with its declaration. Direct access method is important for many applications, for example database system.

3. Indexed Access:-In this method, an index is created for the file. This index contains pointer for various blocks of a file, just like an index in the back of the book. If we want to find a record of a file, first the index is searched and then the pointer from index is used to access that file. In this way, a required record is found. This access method is a slight modification of the direct access method. It is in fact a combination of both the sequential access as well as direct access. The main concept is to access a file direct first and then sequentially from that point onwards. This access method involves maintaining an index. The index is a pointer to a block. To access a record in a file, a direct access of the index is made. The information obtained from this access is used to access the file. For example, the direct access to a file will give the block address and within the block the record is accessed sequentially. Sometimes indexes may be big. So hierarchies of indexes are built in which one direct access of an index leads to info to access another index directly and so on till the actual file is accessed sequentially for the particular record. The main advantage in this type of access is that both direct and sequential access of files is possible.

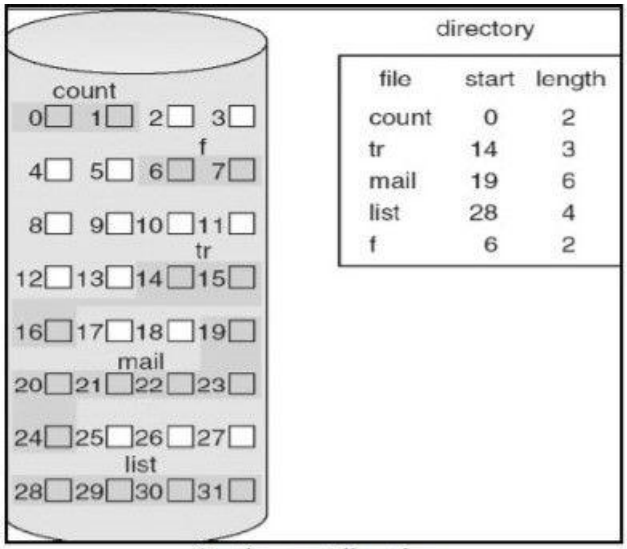
IV. ALLOCATION METHODS OF FILE

Allocation refers to the process of assigning secondary storage space in files. The files should be allocated space in such a manner so that disk space is utilized effectively and files can be accessed quickly. The allocation method is responsible for mapping a file's logical blocks into the actual physical blocks on the secondary storage device. In most operating systems, the size of a physical block is a power of 2 between 512 and 4096. There are three major methods of allocating disk space to files: 1. Contiguous allocation

2. Linked allocation
3. Indexed allocation

1. Contiguous Allocation:- The contiguous allocation method requires each file to occupy a set of contiguous address on the disk. Disk addresses define a linear ordering on the disk. Notice

that, with this ordering, accessing block b+1 after block b normally requires no head movement. When head movement is needed (from the last sector of one cylinder to the first sector of the next cylinder), it is only one track. Thus, the number of disk seeks required for accessing contiguous allocated files in minimal, as is seek time when a seek is finally needed. Contiguous allocation of a file is defined by the disk address and the length of the first block. If the file is n blocks long, and starts at location b, then it occupies blocks b, b+1, b+2, …, b+n-1. The directory entry for each file indicates the address of the starting block and the length of the area allocated for this file. The difficulty with contiguous allocation is finding space for a new file. If the file to be created is n blocks long, then the OS must search for n free contiguous blocks. First-fit, best-fit, and worst-fit strategies (as discussed in Chapter 4 on multiple partition allocation) are the most common strategies used to select a free hole from the set of available holes. Simulations have shown that both first-fit and best-fit are better than worst-fit in terms of both time storage utilization. Neither first-fit nor best-fit is clearly best in terms of storage utilization, but first-fit is generally faster. These algorithms also suffer from external fragmentation. As files are allocated and deleted, the free disk space is broken into little pieces. External fragmentation exists when enough total disk space exists to satisfy a request, but this space not contiguous; storage is fragmented into a large number of small holes. The operating system that uses contiguous allocation is IBM VM/CMS. For example file count starts from 0 and length is 2 so end is 1, file tr starts from 14 and length is 3 so end is 16, file mail is starts from 19 and end 24, file list is starts from 28 and ends with 31 and file f is starts from 6 and end with 8.



Contiguous Allocation

Advantages of Contiguous Allocation:

- It is simple to implement because keeping track of where a file's blocks are reduced to remembering only one number.
- Performance is good because entire file can be read from the disk in a single operation.
- In this scheme, number of disk seeks required for accessing the file is minimal. Disk addresses define linear ordering on the disk, accessing block b+1 after block b requires

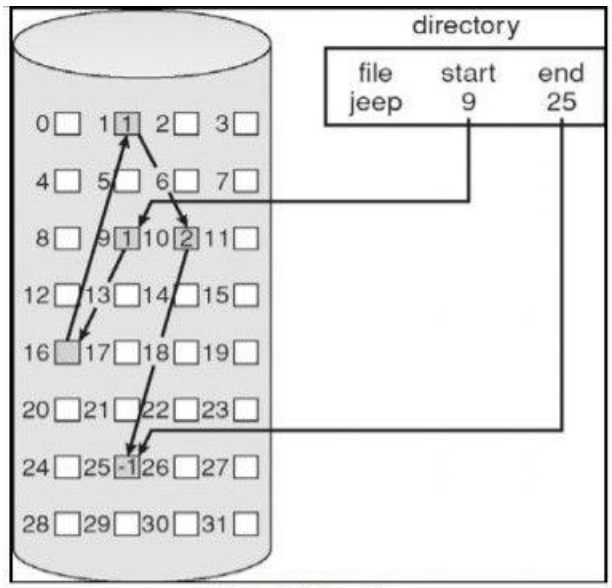
no head movement. As a result number of disk seeks required for a file s are less.

- It is the best from the point of view of the individual sequential file.

Disadvantages of Contiguous Allocation:

- This method from the problem of external fragmentation. As files are allocatd and deleted, the free disk space is broken into little pieces. External fragmentation exists whenever free space is broken into chunks. However, compaction be applied as a solution to this problem, because compaction of the disk is expensive.

2. Linked Allocation:- The problems in contiguous allocation can be traced directly to the requirement that the spaces be allocated contiguously and that the files that need these spaces are of different sizes. These requirements can be avoided by using linked allocation. In linked allocation, each file is a linked list of disk blocks. The directory contains a pointer to the first and (optionally the last) block of the file. For example, a file of 5 blocks which starts at block 9, might continue at block 16, then block 1, block 10 and finally block 25. Each block contains a pointer to the next block and the last block contains a NIL pointer. The value -1 may be used for NIL to differentiate it from block 0. With linked allocation, each directory entry has a pointer to the first disk block of the file. This pointer is initialized to nil (the end-of-list pointer value) to signify an empty file. A write to a file removes the first free block and writes to that block. This new block is then linked to the end of the file. To read a file, the pointers are just followed from block to block. There is no external fragmentation with linked allocation. Any free block can be used to satisfy a request. Notice also that there is no need to declare the size of a file when that file is created. A file can continue to grow as long as there are free blocks. Linked allocation, does have disadvantages, however. The major problem is that it is inefficient to support direct-access; it is effective only for sequential-access files. To find the ith block of a file, it must start at the beginning of that file and follow the pointers until the ith block is reached.



Linked Allocation

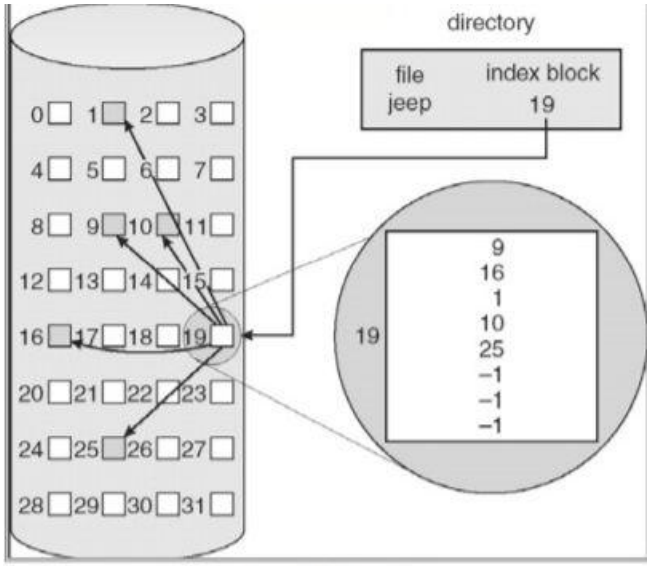
Advantages of Linked Allocation:

- Unlike contiguous allocation every free disk block can be utilized.
- It does not suffer from the problem of external fragmentation. 🎬 There is no need to declare the size of a file at the time of its creation. A file can grow as long as free blocks are available.
- There is no need to perform the compaction.

Disadvantages of Linked Allocation:

- Linked allocation can be used effectively only for sequential files. It is inefficient to support a direct access. In order to find the ith block of a file, we must start at the beginning of that file and follow the pointers until we get to the ith block. Each access to a pointer disk read and a disk seek.
- Pointer takes up space in each disk block. It consumes some portion of a block that can be used for storing information.

3. Indexed Allocation: - The indexed allocation method is the solution to the problem of both contiguous and linked allocation. This is done by bringing all the pointers together into one location called the index block. Of course, the index block will occupy some space and thus could be considered as an overhead of the method. In indexed allocation, each file has its own index block, which is an array of disk sector of addresses. The ith entry in the index block points to the ith sector of the file. The directory contains the address of the index block of a file. To read the ith sector of the file, the pointer in the ith index block entry is read to find the desired sector. Indexed allocation supports direct access, without suffering from external fragmentation. Any free block anywhere on the disk may satisfy a request for more space. For example set the index value is 19 all blocks are linked to that index values provide the references to all blocks. Index method follow the direct method using index block.



Indexed Allocation

Advantages of Indexed Allocation:

- It is the most popular from of file allocation and support both sequential and direct access to the file.
- Any free block on the disk can be used for allocation.
- Allocation of space an the basis of individual block eliminates external fragmentation.
- Allocation of space on the basis of variable size portions improves locality.

Disadvantages of Indexed Allocation:

- If the index block is small, it will not be able to hold enough pointers for a large file.
- The entire index or table will have to be kept in main memory for all the times to make it work.
- Looking up for an entry in a large index is a time consuming process.

CONCLUSION

This paper discusses how the Modify-on-Access file system efficiently extends the capabilities of conventional file systems. It demonstrates how an active file system can simplify both applications and system usage by performing computations on behalf of processes. Furthermore, the paper describes the structure of files and directories. The file system is the first component of a suite of system software designed for a collaborative memory system in which intelligent peripheral devices collaborate with a host processor to accomplish tasks. These implementations are similar to the Active Page and Active Disk simulations described in related work. This provides an extensible environment in which a privileged user implements common, time-critical operations within the kernel and an unprivileged user safely implements user-defined operations outside of the kernel.

REFERENCES

- https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/11_FileSystemImplementation.html
- https://en.wikipedia.org/wiki/File_system
- http://shodhganga.inflibnet.ac.in/bitstream/10603/9868/6/conclusion. pdf
- http://www.tutorialspoint.com/operating_system/os_file_system.html
- http://web.cs.wpi.edu/~cs3013/c07/lectures/Section10- File_Systems.pdf
- http://www.tutorialspoint.com/operating_system/os_file_system.htm
- http://zerofiles.8k.com/fileaccess.html
- https://en.wikipedia.org/wiki/Access_method

- https://www.geeksforgeeks.org/file-allocation-methods/

Windows and Linux Based File System

Tanmay Bansal
Vellore institute of technology
India
tanmay.bansal2019@gmail.com

Abstract-With the rapid development of information technology, computer information security issues are becoming more and more concerned. At present, the market share of Windows operating system exceeds 92%. FAT32 and NTFS are the popular file systems on Windows operating system.
File system of Linux operating system can conserve and manage a lot of important file information. Mining and analyzing the useful data of the Linux operating system have become important means and research directions of computer forensic analysis. In this paper, after the detailed analysis and research of storage principle of linux file system, the object-oriented method is proposed to design the parsing platform of Linux file system.

Keywords—Windows file systems; Linux file system; Ext2; NTFS; FAT

I. INTRODUCTION

With the rapid development of information technology, computers play an increasingly important role in people's work and life. In computing, file system controls how data is stored and retrieved. In other words, it is the method and data structure that an operating system uses to keep track of files on a disk or partition. It separates the data we put in computer into pieces and gives each piece a name, so the data is easily isolated and identified. Without file system, information saved in a storage media would be one large body of data with no way to tell where the information begins and ends.

II. LINUX FILE SYSTEM

The file system, one of the most important ingredients of Linux OS, is used to manage and store the document which is a collection of data, not only contains the data but aIso the structure. Ext2,the basic file system of Linux OS, divides the logic partitions of equipment occupied into many data block groups, which has contained some descriptors and data blocks about the entire file system. The following diagram shows the physical structure of Linux file system.

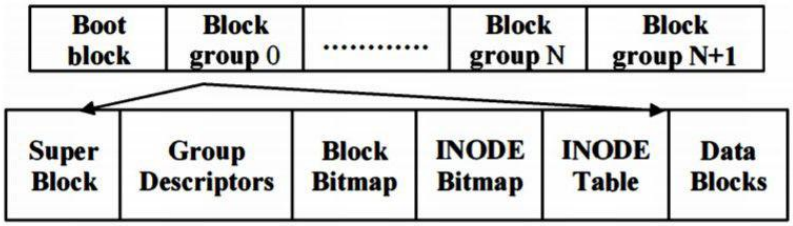


Fig.1 The physical structure of Linux file system

A Ext2 Storage Model

1) Super Block

The Super Block contains important information which is crucial to the booting of the Linux OS, thus backup copies are made in multiple block groups in the file system. However, typically only the first copy of it, which is found at the first block of the file system, is used in the booting. Ext2's super block stored in the structure of Ext2_super block, mainly including the following important information :

• Magic Number: Hook-Programs check the value of magic number to determine whether it's the Ext2 file system or not. The correct value is OxEF53.

• Revision Level: It contains the version number and sub-version number of the file system, which hook programs can be used to determine whether this file system supports some specific file-system-functions or not.

• Mount Count & Maximum Mount Count: By using the Maximum Mount Count to determine whether the system should take a comprehensive check or not. Every time the file system mounted, Mount Count would increase the value of 1.

• Block Group Number: The number of data block group which contains the Super Block.

• Block Size: The data block size of file system, usually be an integer multiple of 512, such as 1024 bytes.

• Blocks per Group: The number of blocks in each group. It is fixed when the file system was created.

• Free Blocks: The number of blocks which are free in the file system.

• Free Inodes: The number of inodes which are free in the file system.

• First Inode: The first index node number of file system; in Ext2, the first inode is the entrance of'/" directory.

2) Group Descriptors

Each data block group has a group descriptor for describing its data structure, just like the super block, each data block group must have a copy of data block group descriptors. Group descriptor table is formed by a group of descriptors, which are placed together. After copying the Super Block, Each data block group contains the entire group descriptor table. The group descriptor stored in the structure of Ext2 group_desc. The following Table presents what data block group descriptor contains.

TABLE I. GROUP DESCRIPTORS

Data block bitmap	The ID of data block that include the allocation bitmap of data block
Inode bitmap	The ID of data block that include the allocation bitmap of inode
Inode table	The ID of first data block that include the inode table
Free block counter, Free INODE counter, Used directory counter	

Linux File System manages disk blocks and Inodes with bitmap, which contains block bitmap and node bitmap. Data block bitmap occupied the space as big as the size of a data block, each bit of which keeps a record of the usage of the data block: the value of the bit is one, which means the disk blocks is available; while the value is zero, it means occupied. The Inode bitmap has the same size of a data block, playing the similar role of the block bitmap: the value of the inode is one, which means the corresponding inode is available; while the value is zero, it means unavailable .

3) Ext2 Inode

The Inode is the foundation of all in Ext2 file system, each file and directory has a different inode, which is kept in the inode table. There is an inode bitmap in the data block group; it's used to record the system's allocated and unallocated inodes.

4) Ext2 Directory Structure

Ext2 realize directory with a special structure, which put the file name and the corresponding inode together. This kind of directory structure is stored in the structure of Ext2 _ dir _entry-2; the main fields of this structure as follows:

- inode: inode number
- rec len: the length of directory entry
- name_len: the length of file name
- file_type: file type
- name: file name

The length of the Ext2 Directory Structure is variable. The last name field of the structure is Variable-length array of Ext2 NAME LEN characters, that's why it varies. Rec_len field is able to point to the next valid directory entry, because you can get the start address of the next valid directory entry with the start address of the current directory entry and Rec _len field as an offset. If the directory entry of the inode field is set to zero with an appropriate increase in the former a valid directory entry Rec _len field, you can delete the directory entry.

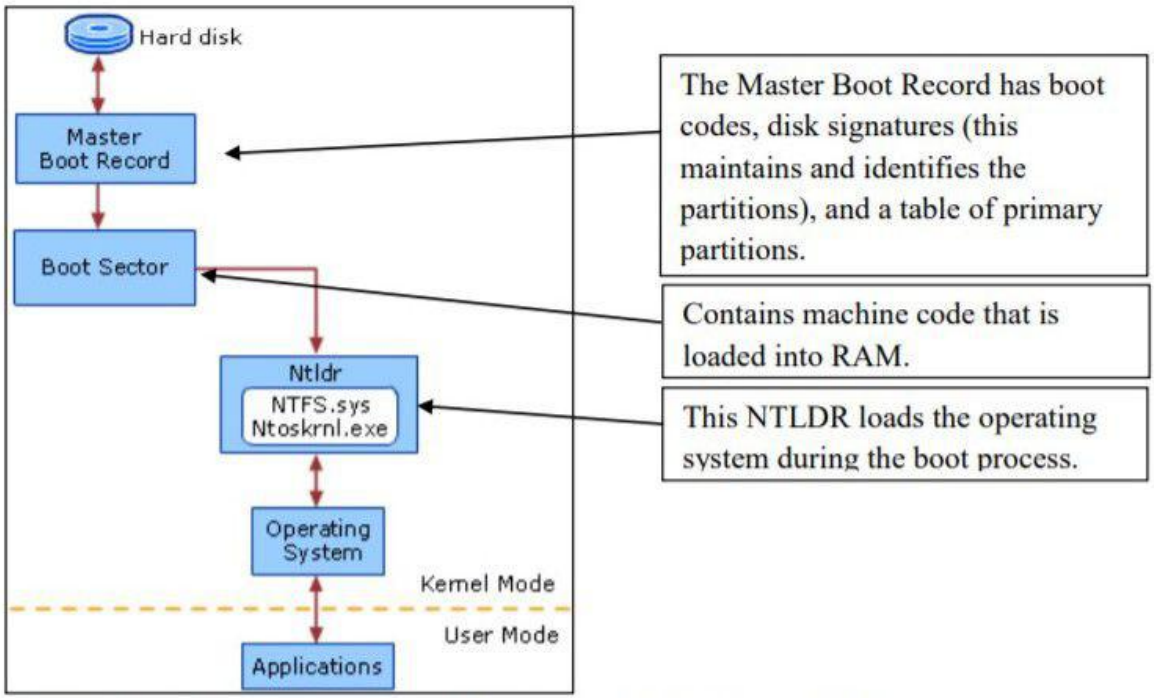
II. WINDOWS FILE SYSTEM

To expand on the book analogy, just as books can divide into sections and chapters, so can the file system be organized into data categories. There are five main existing categories which are file system, content, metadata, file name, and application. Generally, the five categories are able to be applied to a majority of the file systems, though this model must be applied loosely to the FAT file system.

Figure 3: FAT Structure

The file system category can tell you where data structures are and how big the data structures are. This is the general information of the file system. The content category has the data that describes the actual content of the file and generally contains the majority of the file data. The content category is divided into virtual containers, which are usually the clusters or blocks of a hard drive. The metadata category describes and holds the, in layman's terms, "data about data". In other words, the metadata is the data that describes the file data. The location, size, time and date stamps, and access control is all recorded in the metadata category. The file name category is responsible for giving a name to each file. The file name acts as an address for the file. Rather than the user having to remember the address for the file, the file name takes the place of the numbered code, just as a social security number numerically represents a person's name. Finally, the last category is the application category. The application category is not necessary for the organization or reading and writing of the files, but it is solely responsible for the special features in a file. An example of a special feature would be user quota statistics. Often the application category is not even utilized; this is the case for the FAT file system.

All of the components of these file systems have the potential to provide forensic evidence in an investigation. Some of the characteristics are helpful to an investigation and some can hinder the investigation due to their properties or method of operation. Digital evidence submitted into court will need all of the metadata possible to support or deny a claim. For instance, metadata can identify whether an action was human or computer and determine whether something was a mistake, misunderstanding, or on purpose. Metadata can be used to investigate fraud, abuse, and system failures. It can also help establish elements such as causation, timing, extent of knowledge or mens rea, which means guilty mind. Metadata can reveal information about the creation, authorship, history, and intent of documents and files.



Lets differentiate and compare two file systems: NTFS (New Technology File System) and FAT (File Allocation Table), in seven areas. The seven areas are key structures, storage mechanisms, file names, directories, file date and time, file deletion, encryption. The forensic implications of those areas will be discussed after each section. FTK Imager, a forensic extraction tool, will be utilized to give a visual of these differences between the file systems. By understanding the differences between these two file systems, it will be much easier to navigate and its use a forensic tool will be elevated. NTFS is a relatively newer file system, beginning with Windows NT and 2000, and has brought in many new features, including better metadata support and advanced data structures.2 Some added features to NTFS are larger file size, large volume size, last accessed times for files, data access and organization efficiency.23 FAT systems were originally used in DOS and Windows versions prior to windows XP. The "32" in FAT refers to the 32-bit numbers that represent the cluster values, which means that the table entry can have a maximum value of 232 values. Even though the FAT operating system is not utilized in many newer hard drives, it is still often used as a default file system in removable media and storage devices, as well as computers with multiple operating systems. FAT is good for these types of media because it is a very ubiquitous and versatile file system. FAT can also be easily joined with random operating systems, which is why the file system is simplistic when compared to NTFS.

REFERENCES:

- https://ieeexplore.ieee.org/document/737976
- https://ieeexplore.ieee.org/document/7113569
- https://ieeexplore.ieee.org/document/6916481
- https://ieeexplore.ieee.org/document/9110074
- https://research.cs.cornell.edu/projects/Quicksilver/public_pdfs/File%20System%20Usage.pdf
- https://www.marshall.edu/forensics/files/RusbarskyKelsey_Research-Paper-Summer-2012.pdf
- https://linuxexplore.com/2012/10/01/linux-file-system-and-windows-file-system-difference/
- https://www.geeksforgeeks.org/compare-file-system-in-windows-and-linux/

Comparative Study of File Allocation Methods

Pranav Chaudhary(18BEC0902)
B. Tech Electronics and communication Engineering
Vellore institute of technology, Vellore
Tamil Nadu, India
pranav.chaudhary2018@vitstudent.ac.in

Tanmay Bansal(19BCE0421)
B. Tech Computer Science and Engineering
Vellore institute of technology, Vellore
Tamil Nadu, India

1. Abstract: -

Any existing data processing system may have a for example-File Allocation Table which it uses to store files in an organized and systematic way. These systems are categorized by based on the speed and efficiency with which it allocates data. In the following report, a brief about the different types of memory allocations it has and a comparative study of all the file allocation methods. Along with blocks table and figures.

2. Introduction: -

File system is the most important and visible part of an operating system. Managing space on the secondary storage, which includes keeping track of both the memory/file blocks allocated to files and the free block available for allocation. For that we require an effective utilization of space and fast access of files all these points are considered. For this, the operating system should keep track of which disk blocks are free and which disk blocks are allocated. In this report, we learn how disk blocks are allocated to files (file allocation methods). In spite of that, many considerations are similar to both environments, particularly, contiguous and non-contiguous allocation of the files. Each method has its advantages and disadvantages which will be discussed and compared throughout.

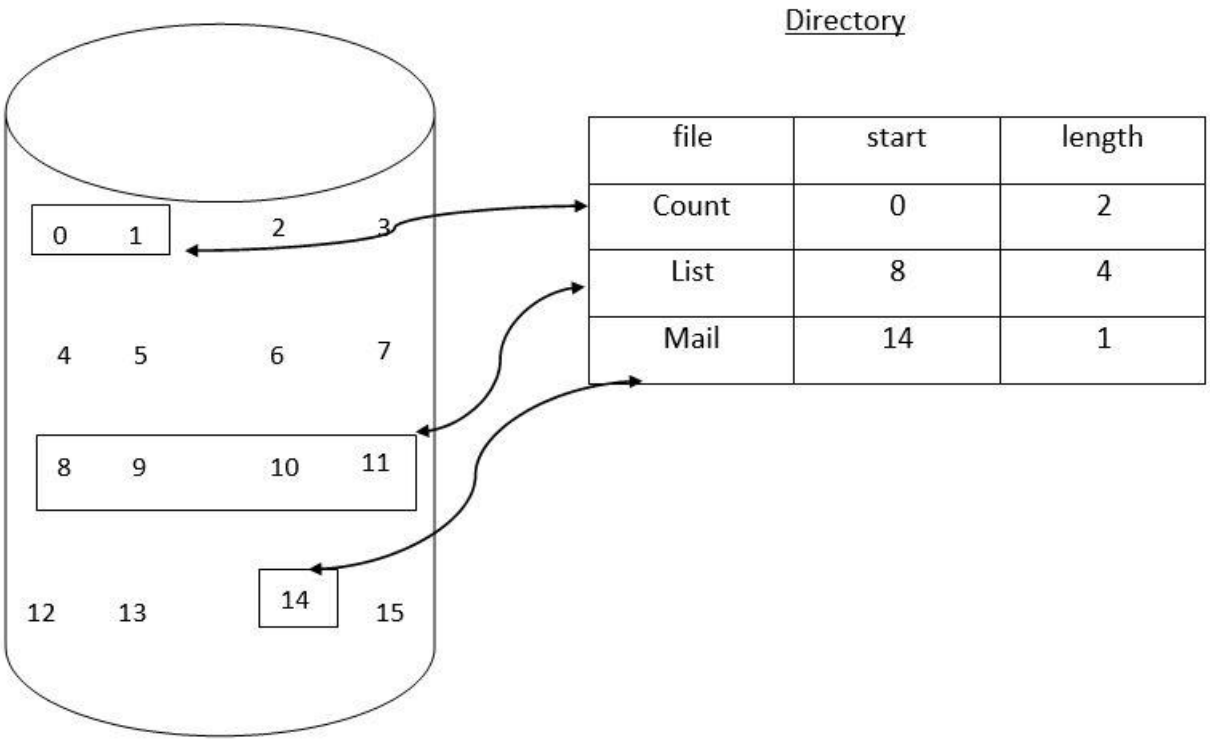
File Allocation method is used for effectively utilize disk space available by accommodating space for files. Operating system allocates the disk space for the files by using different approaches. Which are divided majorly into three broad categories: Contiguous Allocation, Linked Allocation and Indexed Allocation. The main objective to adapt these 3 approaches is to make sure their efficient utilization of disk space and speedy access to the files. These three methods use distinct methods to effectively use disk space which has its own pros and cons.

Comparative Study of Three Approaches: -

2.1 Contiguous allocation method:

If a file needs t blocks and is given a block x as its beginning point/address, then the block acquired by the file will be x, x+1x+2, … x+t-1 in a contiguous form. So, for any file we have to provide start point and its allocated length. This will be great for a simple implementation and also for a sequential file.

3.File allocation methods: -



Contiguous allocation method

As shown in the above figure, 'count' file is stored on disk and the starting block from where the file begins is block number 0 (zero) and the length of the file is 2 blocks long. So, block number 0 and 1 are allocated for the file 'Hello'. Same way allocation for 'Mail' and 'List' file will be done.

This diagram exhibits similar fragmentation problems as in variable memory partitioning. This is because the allocation and deal location could result in regions of free disk space broken into pieces within active space, which is called external fragmentation.

Advantages:

- It is simple to implement because keeping track of where a file's blocks are reduced to remembering only one number.
- Performance is good because entire file can be read from the disk in a single operation.
- In this scheme, number of disk seeks required for accessing the file is minimal. Disk addresses define linear ordering on the disk, accessing block b+1 after block b requires no head movement. As a result number of disk seeks required for a file are less.
- It is the best from the point of view of the individual sequential file.

Disadvantages:

- This method suffers from both internal and external fragmentation. This makes it inefficient in terms of memory utilization.
- Increasing file size is difficult because it depends on the availability of contiguous memory at a particular instance.

Simple basic algorithm: -

Step 1: Start the program.

Step 2: Get the number of files.

Step 3: Get the memory requirement of each file.

Step 4: Allocate the required locations to each in sequential order.

a). Randomly select a location from available location s1= random (100);

b). Check whether the required locations are free from the selected location.

c). Allocate and set flag=1 to the allocated locations.

Step 5: Print the results file no, length, Blocks allocated. Step

6: Stop the program.

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define TB 100
void allocate();
void display();
int BFull();
struct Sequence
{
 char n[30];
 int len;
 int st;
}F[30];
int Table[TB],pos=0,r,i,j,ch,B=0;
char fn[30];
int main()
{
 printf("\n Contiguous File Allocation \n\n");
 do{
 printf("\n\n1.Allocate\n2.Display\n3.Exit");
 printf("\n\nEnter Your choice : ");
 scanf("%d",&ch);
 switch(ch)
 {
 case 1:
 pos++;
 allocate();
 break;

 case 2:
 display();
 break;
 case 3:
 exit(0);
 default:
 printf("\nInvalid Choice ");
 }
 }while(ch!=3);
}
void allocate()
{
```

```
 printf("\nEnter File Name : ");
 scanf("%s",&(F[pos].n));
 printf("\nEnter File Length : ");
 scanf("%d",&(F[pos].len));
 if(BFull())
 {
 pos--;
 printf("\n\nNo Enough Free Space Available \n");
 return;
 }
 while(1)
 {
 i=0;
 r=(rand()%TB+1);
 if(r+F[pos].len-1>TB)
 continue;
 if(Table[r]==0)
 {
 for(i=r+1;i<r+F[pos].len;i++)
 if(Table[i]==1)
 break;
 }
 if(i==r+F[pos].len)
 break;
 }
 F[pos].st=r;
 for(i=r;i<r+F[pos].len;i++)
 Table[i]=1;
 printf("\n\tFile Allocation Table\n");
 printf("\nFileName\tStart\tLength\n");
 for(i=1;i<=pos;i++)
 {
 printf("\n%s\t\t%d\t%d",F[i].n,F[i].st,F[i].len);
 printf("\n");
 }
}
void display()
{
 printf("\nEnter The File Name : ");
 scanf("%s",fn);
 printf("\nBlocks Allocated Are : ");
 for(i=1;i<=pos;i++)
 {
 if(strcmp(F[i].n,fn)==0)
 {
 for(j=F[i].st;j<(F[i].st+F[i].len);j++)
 printf("--%d--",j);
 break;
 }
 }
}
int BFull()
{
 for(i=1,B=0;i<=pos;i++)
 B=B+F[i].len;
 if(B>TB)
 return 1;
 else
 return 0;
}
```

Output:

```
pranav_malik1@LAPTOP-JM1BDI4J:~$ ./a.out

 Contiguous File Allocation



1.Allocate
2.Display
3.Exit

Enter Your choice : 1

Enter File Name : file1

Enter File Length : 12

       File Allocation Table

FileName        Start   Length

file1           84      12


1.Allocate
2.Display
3.Exit

Enter Your choice : 1

Enter File Name : file2

Enter File Length : 15

       File Allocation Table

FileName        Start   Length

file1           84      12

file2           16      15
```

```
1.Allocate
2.Display
3.Exit

Enter Your choice : 2

Enter The File Name : file2

Blocks Allocated Are : --16----17----18----19----20----21----22----23----24----25----26----27----28----29----30--

1.Allocate
2.Display
3.Exit

Enter Your choice : 2

Enter The File Name : file1

Blocks Allocated Are : --84----85----86----87----88----89----90----91----92----93----94----95--

1.Allocate
2.Display
3.Exit

Enter Your choice : 3
pranav_malik1@LAPTOP-JM1BDI4J:~$
```

2.2 Linked list Allocation Method:

In this method each block will be allocated anywhere throughout the disk. The index contains a pointer to the beginning and the ending block. Each block contains a pointer to its next block involved in the allocation of that file. Works most efficient with sequential files as even without contiguous memory allocation we can access all block from a file.

The diagram shows linked allocation where each block contains the information about the next block.

The directory entry contains the block number of the first block of the file. The table entry indexed by block number contains the block number of the next block in the file. The Table pointer of the last block in the file has an EOF pointer value. This chain continues until EOF (end of file) table entry is encountered.

Linked allocation can be used effectively only for sequential files. It is inefficient to support a direct access. In order to find the ith block of a file, we must start at the beginning of that file and follow the pointers until we get to the ith block. Each access to a pointer disk read and a disk seek. Pointer takes up space in each disk block. It consumes some portion of a block that can be used for storing information.

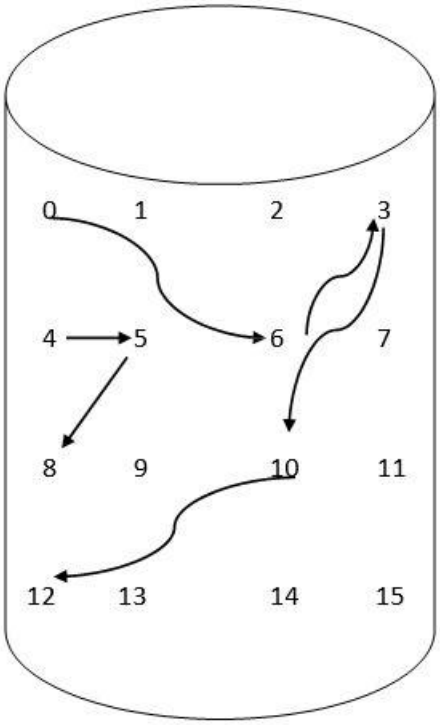
Advantages:

- Unlike contiguous allocation every free disk block can be utilized.
- It does not suffer from the problem of external fragmentation. There is no need to declare the size of a file at the time of its creation. A file can grow as long as free blocks are available.
- There is no need to perform the compaction.

Disadvantages:

- Linked allocation can be used effectively only for sequential files. It is inefficient to support a direct access. In order to find the ith block of a file, we must start at the beginning of that file and follow the pointers until we get to the ith block. Each access to a pointer disk read and a disk seek.
- Because the file blocks are distributed randomly on the disk, a large number of seeks are needed to access every block individually. This makes linked allocation slower.

Pointer takes up space in each disk block. It consumes some portion of a block that can be used for storing information.



Directory

File	Start	end
System	0	13
Dial	4	8

Linked list Allocation Method

Simple basic algorithm: -

Step 1: Start the Program

Step 2: Get the number of files.

Step 3: Allocate the required locations by selecting a location randomly Step 4:

Check whether the selected location is free.

Step 5: If the location is free allocate and set flag =1 to the allocated locations. Step 6: Print

the results file no, length, blocks allocated.

Step 7: Stop the execution

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define TB 100
void allocate();
void display();
int BFull();
struct Link
{
 char n[30];
 int len;
 int st;
 struct node
 {
 int index;
 struct node *next;
 }*Start,*current,*newnode,*temp;
}F[30];
int Table[TB+1],pos=0,r,i,j,ch,B=0;
char fn[30];
int main()
{
 printf("\n Linked File Allocation \n\n");
 do{
 printf("\n\n1.Allocate\n2.Display\n3.Exit");
 printf("\n\nEnter Your choice : ");
 scanf("%d",&ch);
 switch(ch)
 {
 case 1:
 pos++;
 allocate();
 break;
 case 2:
 display();
 break;
 case 3:
 exit(0);
 default:
 printf("\nInvalid Choice ");
 }
 }while(ch!=3);
}
void allocate()
{
 printf("\nEnter File Name : ");
 scanf("%s",&(F[pos].n));
 printf("\nEnter File Length : ");
 scanf("%d",&(F[pos].len));
 F[pos].Start=NULL;
 if(BFull())
 {
 pos--;
 printf("\n\nNo Enough Free Space Available \n");
 return;
 }
 for(i=1;i<=F[pos].len;i++)
 {
 F[pos].newnode=(struct node *)malloc(sizeof(struct node));
```

```
while(1)
{
r=rand()%TB+1;
if(Table[r]==0)
{
F[pos].newnode->index =r;
F[pos].newnode->next=NULL;
if(F[pos].Start==NULL)
{
F[pos].Start=F[pos].newnode;
F[pos].current=F[pos].newnode;
}
else
{
F[pos].current->next=F[pos].newnode;
F[pos].current=F[pos].newnode;
}
Table[r]=1;
break;
}
}
}
F[pos].st=F[pos].Start->index;
printf("\n\tFile Allocation Table\n");
printf("\nFileName\tStart\tEnd\tLength\n");
for(i=1;i<=pos;i++)
{
printf("\n%s\t\t%d\t%d\t%d",F[i].n,F[i].st,F[pos].current->index,F[i].len);
printf("\n");
}
}
void display()
{
printf("\nEnter The File Name : ");
scanf("%s",fn);
printf("\nBlocks Allocated Are : ");
for(i=1;i<=pos;i++)
{
if(strcmp(F[i].n,fn)==0)
{
F[i].current=F[i].Start;
while(F[i].current)
{
printf(">--%d-->",F[i].current->index);
F[i].current=F[i].current->next;
}
break;
}
}
if(i==pos+1)
{
printf("\n\nNo File Found\n");
}
}
int BFull()
{
for(i=1,B=0;i<=pos;i++)
B=B+F[i].len;
if(B>TB)
return 1;
else
```

```
 return 0;
}
```

Output:

```
pranav_malik1@LAPTOP-JM1BDI4J:~$ ./a.out

 Linked File Allocation



1.Allocate
2.Display
3.Exit

Enter Your choice : 1

Enter File Name : file1

Enter File Length : 12

       File Allocation Table

FileName        Start   End     Length

file1           84      91      12


1.Allocate
2.Display
3.Exit

Enter Your choice : 1

Enter File Name : file2

Enter File Length : 15

       File Allocation Table

FileName        Start   End     Length

file1           84      91      12

file2           60      23      15
```

```
1.Allocate
2.Display
3.Exit

Enter Your choice : 2

Enter The File Name : file1

Blocks Allocated Are : >--84-->>--87-->>--78-->>--16-->>--94-->>--36-->>--93-->>--50-->>--22-->>--63-->>--28-->>--91-->

1.Allocate
2.Display
3.Exit

Enter Your choice : file2

Enter The File Name :
Blocks Allocated Are : >--60-->>--64-->>--27-->>--41-->>--73-->>--37-->>--12-->>--69-->>--68-->>--30-->>--83-->>--31-->>--24-->>--3-->>--23-->

1.Allocate
2.Display
3.Exit

Enter Your choice : 3
pranav_malik1@LAPTOP-JM1BDI4J:~$
```

2.3 Indexed Allocation Method:

In this method we are using index block with carry all the other pointers involved in the same file. The ith section in the index block contains the disk address of the ith file block. Best method for allocation as it supports both sequential and direct address.

As we can see in the below diagram two file naming 'system' and 'dial' are created with index allocation and as we can see for accessing the file, we require only the index block which carry all the memory address to access the file.

Advantages:

- It is the most popular from all of file allocation and support both sequential and direct access to the file.
- Any free block on the disk can be used for allocation.
- Allocation of space based on individual block eliminates external fragmentation.

Disadvantages:

- The pointer overhead for indexed allocation is greater than linked allocation.
- For very small files, say files that expand only 2-3 blocks, the indexed allocation would keep one entire block (index block) for the pointers which is inefficient in terms of memory utilization.
- If the index block is small, it will not be able to hold enough pointers for a large file.

- The entire index or table will have to be kept in main memory for all the times to make it work.
- Looking up for an entry in a large index is a time-consuming process.

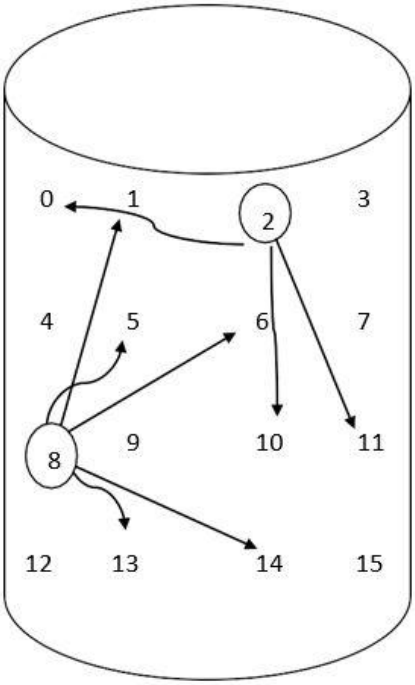
Also, for an increased file size we can easily add new blocks to the index blocks. But for a very large files, a single index block may not sufficient to hold all the pointer values.

We can resolve this issue:

1. Linked scheme: This scheme refers to when two or more index blocks are linked together so that a large pool exists for holding all pointers. Each index block would contain a pointer or the address to the next index block holding the next set of file blocks pointers.

2. Multilevel index: In this policy, a multilevel hierarchy between index blocks exists. Each index block would point to the next and so on. This policy can be extended to 3 or more levels of index blocks, depending on the size of the file.

Combined Scheme: In this scheme, a special block called the Information Node (inode) contains data about the file such as the name, size, etc. Also, the inode block is used to store the disk block addresses which contains the actual file through single and double indirect block.



Directory

File	Index block
system	8
dial	2

Indexed Allocation Method

Simple basic Algorithm: -

Step 1: Start the Program

Step 2: Get the number of files.

Step 3: Get the memory requirement of each file.

Step 4: Allocate the required locations by selecting a location randomly. Step 5:

Print the results file no, length, blocks allocated.

Step 6: Stop the execution.

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define TB 100
void allocate();
void deallocate();
void display();
int BFull();
struct Indexed
{
 char n[30];
 int len;
 int indexblock;
 int *IB[TB];
}F[30];
int Table[TB+1],pos=0,r,i,j,ch,B=0,indexblock,k=0;
char fn[30];
```

```
int main()
{
 printf("\n Indexed File Allocation \n\n");
 do{
 printf("\n\n1.Allocate\n2.Display\n3.Exit");
 printf("\n\nEnter Your choice : ");
 scanf("%d",&ch);
 switch(ch)
 {
 case 1:
 pos++;
 allocate();
 break;
 case 2:
 display();
 break;
 case 3:
 exit(0);
 default:
 printf("\nInvalid Choice ");
 }
 }while(ch!=3);
}
void allocate()
{
 printf("\nEnter File Name : ");
 scanf("%s",&(F[pos].n));
 printf("\nEnter File Length : ");
 scanf("%d",&(F[pos].len));
 if(BFull())
 {
 pos--;
 printf("\n\nNo Enough Free Space Available \n");
 return;
 }
 F[pos].indexblock=rand()%TB+1;
 Table[F[pos].indexblock]=1;
 for(i=1;i<=F[pos].len;i++)
 {
 while(1)
 {
 r=rand()%TB+1;
 if(Table[r]==0)
 {
 F[pos].IB[i]=&Table[r];
 Table[r]=r;
 break;
 }
 }
 }
 printf("\n\tDirectory\n");
 printf("\nFileName\tIndexBlock\n");
 for(i=1;i<=pos;i++)
 {
 printf("\n%s\t\t%d",F[i].n,F[i].indexblock);
 printf("\n");
 }
}
```

```
void display()
{
 printf("\nEnter The File Name : ");
 scanf("%s",fn);
 printf("\nBlocks Allocated Are : \n");
 for(i=1;i<=pos;i++)
 {
 if(strcmp(F[i].n,fn)==0)
 {
 printf("\n\t IndexBlock\n");
 printf("\n\t %d\n",F[i].indexblock);
 printf("\t------------");
 for(j=1;j<=F[i].len;j++)
 {
 printf("\n\t| %d\t |\n",*F[i].IB[j]);
 }
 printf("\t------------");
 break;
 }
 }
 if(i==pos+1)
 {
 printf("\n\nNo File Found\n");
 }
}
int BFull()
{
 for(i=1,B=0;i<=pos;i++)
 B=B+F[i].len;
 if(B>TB-pos-k)
 return 1;
 else
 return 0;
}
```

Output:

```
pranav_malik1@LAPTOP-JM1BDI4J:~$ ./a.out

 Indexed File Allocation



1.Allocate
2.Display
3.Exit

Enter Your choice : 1

Enter File Name : file1

Enter File Length : 12

        Directory

FileName         IndexBlock

file1            84


1.Allocate
2.Display
3.Exit

Enter Your choice : 1

Enter File Name : file2

Enter File Length : 15

        Directory

FileName         IndexBlock

file1            84

file2            64
```

```
1.Allocate
2.Display
3.Exit

Enter Your choice : 2

Enter The File Name : file1

Blocks Allocated Are :

         IndexBlock

         84
        ------------
        | 87     |
        | 78     |
        | 16     |
        | 94     |
        | 36     |
        | 93     |
        | 50     |
        | 22     |
        | 63     |
        | 28     |
        | 91     |
        | 60     |
        -------------
```

```
1.Allocate
2.Display
3.Exit

Enter Your choice : file2

Enter The File Name :
Blocks Allocated Are :

         IndexBlock

         64
        ------------
        | 27     |

        | 41     |

        | 73     |

        | 37     |

        | 12     |

        | 69     |

        | 68     |

        | 30     |

        | 83     |

        | 31     |

        | 24     |

        | 3      |

        | 23     |

        | 59     |

        | 70     |
        ------------

1.Allocate
2.Display
3.Exit

Enter Your choice : 3
pranav_malik1@LAPTOP-JM1BDI4J:~$
```

3. Performance Analysis: -

Among the 3 types of file allocation methods, the best one is supposed to be the indexed file allocation method.

- Indexed File Allocation overcomes all the shortcomings caused by the Linked Allocation method and continuous allocation method.

- Indexed File Allocation method supports direct accessing which makes it faster and more efficient.

- In the Indexed File Allocation Method, source code and outputs received that the File Allocation Table is a great method in order to store files and have redundancy.

- It makes deleting files and updating files easier.

4. Result: -

After comparing the different file allocation methods, we came to the conclusion that the best one is indexed file allocation as it supports direct accessing which is faster and efficient, overcomes the disadvantages of both contiguous and indexed file allocation methods. Apart from this, a file allocation table has been made which is the best method to store files. Implementing a file allocation table makes deleting and updating quicker and more efficient.

References: -

1. Disk Space Management Methods Ramakrishna ChowdaryM.C.A. Department Rao & Naidu Engineering College [RNEC] Ongole, Prakasam Andhra Pradesh – India.

2. Cluster Allocation strategies of the ExFAT and FAT file systems: A comparative study in embedded storage systems Keshava Munegowda 1 Dr. G T Raju 2 Veera Manikandan Raju.

3. Directory Structure and File Allocation Methods Mandeep Kaur, Sofia Singh, Rupinder Kaur Assistant Professor, PG Department of Computer Science and Applications, GHG Khalsa College Gurusar Sadhar, Ludhiana, Punjab, India.

4. Role of File System in Operating System Faheem Hafeez College of information Technology, University of Punjab, Jhelum, Pakistan Email: Enggfahim44@gmail.com.

5. e-PG Pathshala Computer Science Operating Systems File Allocation Methods Module No: CS/OS/35 Quadrant 1 — e-text.

HDFS,HIVE,MapReduce,YAFFS2

Distributed File Systems and Mobile File System

Neel Rakesh Choksi(19BCE0990)
B.Tech Computer Science and Engineering
Vellore Institute Of Technology,Vellore
Tamil Nadu, India
neelrakesh.choksi2019@vitstudent.ac.in

***Abstract*—I have done more research after writing the previous report about HDFS and YAFFS. Hadoop Distributed File System does all its jobs using the MapReduce analogy. Hive is built on top of MapReduce .Hive provides a SQL environment to deal with tabular data and performs all the complex mapreduce tasks. .Along with that I have moved ahead in discovering YAFFS2. I have understood and explained the concept of wear levelling and garbage collection in YAFFS2 during the analysis.**

Keywords—HDFS, YAFFS2 , MapReduce, Hive,Wear levelling, Garbage Collection

I. INTRODUCTION

This is the final report for the project on Distributed File System and Mobile File System.I have discovered more areas in the HDFS . I have set up Hadoop on pseudo distributed mode. It takes in data into the HDFS through a MapReduce program (written in the Maeven environment). This data is further transferred to the derby database in Hive and data is analysed using HiveQL queries .I have also shown the analysis of YAFFS2 mobile file system.

II. INSIGHTS FROM PREVIOUS REPORT-THEORY

There are many types of file systems both for mobile and for computers. Currently Android mobiles have ext4 file system which is a linux file system. Currently Linux distributions use ext* ,XFS,JFS(Journaling File System),btrfs(b-tree file system). Windows currently uses NTFS(NT File System) .Currently mac os uses APFS(Apple File System).

While discovering about HDFS I found out that Hive is the data warehouse that can import data from HDFS and can analyse it and store it back to HDFS. So I went ahead with it.I found out that HiveQL is used in Hive to analyse the data . Further I found out that Hive was built to simplify and Optimize the programs written in MapReduce .So I found out more about MapReduce. It can be executed by writing a .jar file in the HDFS configuration . This .jar file can be created using a Java API . For YAFFS2 I found out more about its structure. It performs wear levelling and garbage collection to store files.I also found more about Flash memory.

III. HADOOP

A. Big Data

Traditional computing techniques are not enough to process large datasets. Social media generates a lot of data.550 new social media users are registered in each minute.According to the survey in Jan 2020,our total population is 7.75 Billion, we have 5.19 Billion unique mobile users . We have 4.54 Billion internet users and 3.80 Billion active Social media users. This data has to be computed and the companies that own the data centres have to manage this data.The amount of data generated is increasing by an enormous amount. 90 percent of the currently available data was generated in the past few years.Every second there are 43 thousand google searches, 7 thousand tweets, 2 million emails, 83 thousand youtube videos watched,21 TB of internet traffic .Businesses need to analyse their data in order to generate better revenue, find their target audience and show them advertisements based on their need[29].

There are different types of big data namely Social media, Structured Data, Customer Feedback Surveys , Text Messages and emails ,Call center and ATM data.Some more examples are black box data in aeroplanes and helicopters and jets . The black box stores microphone voices , performance statistics of the aeroplane and also the information about the plane. Social Media dataset has a lot of user information . Stock Exchange data has a lot of data about companies shares and the history of the rate of the shares. This data is used and analysed by many traders who buy and sell shares according to the performance of the company. The previous data of a particular share helps the traders to analyse and predict the share price in the future.Transportation is evolving .There is a lot of research going on automating the cars. Different types of sensors are used to track the speed, distance and nearby vehicles, humans .These sensors collect data every millisecond , generally the data is log structured . This data is used by the car itself to get habituated with different situations so that self driving cars can be more safe. There are 43 thousand searches on google per second, to accumulate and analyse this huge amount of unstructured data we need a framework[30].

There are different types of big data.The main three types are Structured, Unstructured, and Semi-structured

data.Structured data is the data that is generally collected using traditional relational database management systems. Here the tables are fixed and the columns are also fixed,only one type of data can be collected, there are constraints put on the data so that only apt data is entered into the database.For an instance,the employee information database has fixed entities like the branch , salary ,name, age , employee number.This data is already stored in an order. 20% of the total data is structured data . This data is generally generated by sensors and machines and weblogs. Humans also generate structured data but only when they are doing an operation which involves a database which checks the data before storing it.Unstructured data does not have a clear format for storage. We can store structured data in rows and columns format but unstructured data cannot be stored in this form. 80% of the total data is unstructured in nature . For an instance satellite images,machine generated unstructured data,scientific data and also humans generate a lot of unstructured data such as images and videos on social media , pdf documents and text documents. This document is also a type of unstructured data. The third type of data is semi-structured data. It is the most ideal form of data . It is in the form of JSON(JavaScript Object Notation) and XML(eXtensible Markup Language). It cannot be stored in a database but it has many organizational properties with which it can be structured. For instance Spreadsheet files, XML files of an application which defines the structure of an application. Also JSON docs and NoSQL(Not Only SQL) database data items come under semi-structured data[31].

Big data technologies exist for operational and analytical purposes. For operational purposes that is for collection of data the technologies available are Real time Interactive Database, NoSQL databases. These technologies provide operational competencies and velocity. These technologies are used in real time web/mobile/IoT applications which are used by millions of customers. These technologies are mainly used to provide a solution for a problem by giving a digital solution, in turn it gets to collect data from the users, which helps them improve their customer base and expand it. The other type of technologies are the Analytical type which analyses all the collected data and suggests improvement in the service which the company is providing to their customers. This mainly happens offline and not in real time.It is batch-oriented. Batch oriented means that the system takes in a batch of data and executes all the jobs on it and results for that batch.Hadoop is mainly used to achieve this along with some more analytical databases. These are the databases that allow the data to be stored first and then apply constraints and checks on the data. Therefore the important aspects of big data are Volume , Variety, Velocity, Value, Validity. Volume is the huge amount of data that is generated and processed .Variety refers to the different types of data received from different sources. Velocity refers to the high speed accumulation of data. Value refers to the useful data extracted from the huge amount of data. Validity refers to the incompetency and the uncertainty of the data.

To implement these functionalities we have to collect and process these huge datasets. We cannot store and process an infinite amount of data . Google had first started processing data coming to their servers using their Distributed File Systems. Traditional Relational Database Management Systems cannot process such a huge amount of Data.

B. Hadoop

In 2003 Dough Cutting was working on a search engine called nutch as a side project. He wanted it to be scalable since petabytes and terabytes of data had to be processed. It was working on 4 to 5 machines but still with some errors. At that time google had published some research papers regarding their Distributed File System and about their framework called MapReduce. They had given an idea about how they were handling big data. Dough with his partner worked on nutch and they were able to run it on 50-60 machines .Still they wanted it to scale more. At that time Yahoo was having an issue related to processing large amounts of data , so Yahoo approached Dough and let them collaborate and created a new section for distributed file systems from the nutch engine. They called this project Hadoop. Hadoop was able to scale on thousands of machines and they were able to process petabytes of data . Now, hadoop has turned up to be the kernel, an operating system for big data[32]. There are many tools that come along with hadoop that enable you to do complex mapreduce tasks using SQL . Some of the examples are Hive, Spark, and Pig which is a dataflow language, also interactive SQL with impala.It has become the general purpose platform for data processing that scales much better and is much more flexible than any other platform available.And most importantly it is open source. All the source code is available openly.Hadoop solves the problem of storage and processing. There is no limit to store data in Hadoop Distributed File System. MapReduce is a framework that enables parallel processing for faster response.

Now hadoop is maintained by Apache software foundation. Apache Hadoop is an open source framework. Hadoop can easily handle large amounts of data on a lower cost and simple hardware clusters.Hadoop is also scalable and a fault tolerant framework. It is not only a storage system but also it is a framework that allows the processing of data . Hadoop systems are written in java, and horizontal scaling is also possible. Horizontal scaling involves adding new servers rather than increasing the capacity and GPU processing of the same servers(Vertical Scaling). Hadoop can be run on commodity hardware and it is fault tolerant. Commodity hardware is the set of hardware required which has no particular specifications. For an instance to run Oracle database the server machines should have a particular RAM and Processors.

Hadoop is open source technology from Apache Software Foundation .Most of the code is written by Yahoo, IBM and Cloudera. Parallel processing can be achieved using commodity hardware. The cost of the hardware is low hence this system becomes economic. Hadoop is quite popular and captures around 90% of the big data market.Hadoop is scalable and many commodity hardwares can be added easily. If one node goes down then the other can take over, since it

also performs data replication. Data can be stored in three forms namely structured , unstructured , semi-structured[32].

C. Hadoop Ecosystem

I have used Hive for importing the data from HDFS and displaying it in a table format. I have implemented MapReduce programs that help in analysing data.

There are many vendors available for hadoop . Hortonworks, Cloudera, mapreduce ,Microsoft, IBM but I have used vanilla hadoop which has no prebuilt setup which has helped me understand the structure of the hdfs

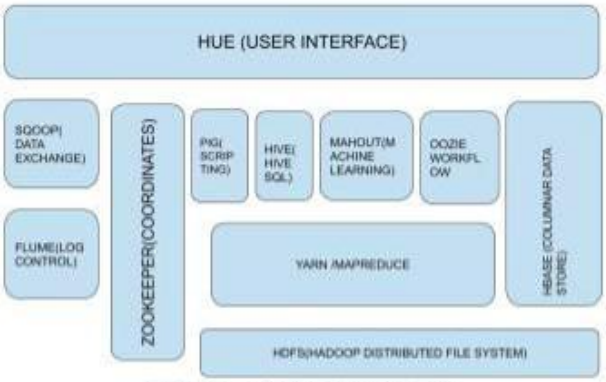


Figure 1. Hadoop Ecosystem

IV. Features of Hadoop

The main features of hadoop include it is cost effective,can be run on large clusters of nodes , used for parallel processing, data is stored in a distributed manner, automatic failover management, data locality optimization ,heterogeneous cluster, scalability.

A. *Cost effective*

Traditional RDBMS(Relational Database Management System) like Oracle is a licensed software and it requires machines with particular specifications to run Oracle database.Whereas Hadoop is totally open source and free of cost . It works on commodity hardware which means that even cheap hardware that fulfills the basic criteria can also run Hadoop.These hardware are bound to fail but hadoop handles hardware failure automatically.

B. Large Cluster of Nodes

Oracle has a limit of 256 machines, Teradata has a limit of 512 machines whereas hadoop has no limit for the number of machines. A large cluster can be set up therefore more computing and storage is achieved.

C. Parallel Processing

Mapreduce is the framework through which hadoop does all its processing. Hadoop can process data in a parallel fashion.

It follows a master slave technique wherein the master node controls the data flow into the slave nodes and processes it in parallel in them, if one data node fails then the respective data is found and fetched from other data nodes since replication is followed by Hadoop.

D.Distributive Data

Data is stored into multiple machines and stored. Data is stored in the form of chunks called blocks . These blocks are stored on the machines in the clusters.

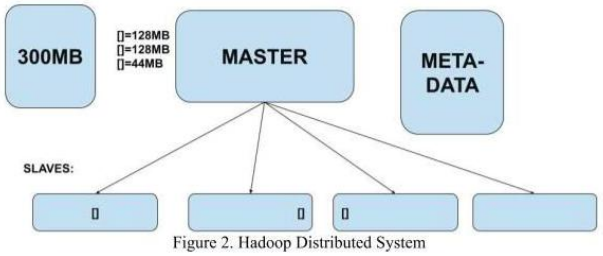


Figure 2. Hadoop Distributed System

E.Automatic Failover Management

By default there are three blocks replicated for the stored data . These three copies of data are stored in a distributed manner on the cluster of machines. For example there is a file of 300 MB . The default block size is 128MB therefore the file will be divided into three blocks and these three blocks get replicated 3 times therefore totally 9 blocks are stored in the HDFS. This replication is determined by the replication factor which can be configured in the cluster .

F.Heterogeneous Cluster

All the machines on a cluster can be configured using different Hardwares and operating systems.In Oracle it is necessary to have a particular hardware and operating system on every machine.

G.Scalability

We can add any number of machines on our cluster and the storage has also no limits. Compared to other systems, hadoop does not require a maintenance mode wherein all the systems are down for sometime.

V. Hadoop Architecture

Hadoop follows a master slave architecture as shown in the figure related to Distributive Data. It has Mapreduce for distributed computation.It has a Distributed File System storage called HDFS(Hadoop Distributed File System) . Yarn framework is available for cluster management and there are other common utilities in the Hadoop architecture.

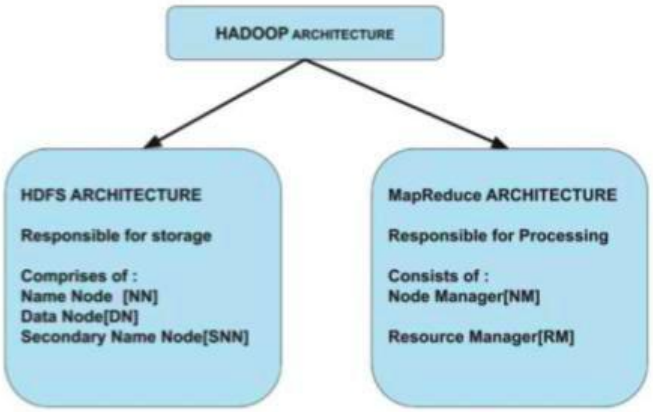


Figure 3. Hadoop Architecture

Hadoop Architecture has five daemons namely the NameNode, DataNode, Secondary NameNode, Resource Manager, Node Manager.

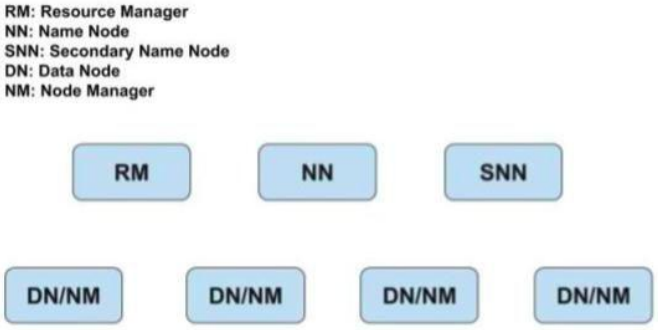


Figure 4. Hadoop Daemons

Data Nodes , Name Nodes and Secondary Namenodes are a part of HDFS and Resource Manager and Node Manager comes under MapReduce. Name Node , Secondary NameNode and Resource Manager are the master daemons and they are generally kept on different machines.Daemon Processes are those processes that run in the background.

Yarn has hadoop's Node Manager and Resource Manager. It is built on top of HDFS to do resource management.YARN stands for Yet Another Resource Negotiator. It is a layer that separates the resource management layer and the processing components layer.Resource Manager handles the client requests . The node managers is a task tracker that accesses resources from the Resource Manager.Node Manager has two components one is the AppMaster and the Container.The App Master manages resource needs of individual applications.Node manager therefore oversees the containers running on the cluster nodes.[30]

When a MapReduce task has to be run, the client requests the Resource Manager which in turn sends the task to Node Manager. Task is executed and the resources required are collected from the MapReduce App master which collects them from the Data Node. The resource manager helps in collecting them from the MapReduceAppMaster.

VI. Hadoop Modes

Hadoop runs on three modes namely

Local/Standalone Mode , Pseudo Distribution Mode, Fully Distribution Mode.It has Java Virtual Machine installed on all the machines to run the java programs.By default Hadoop is installed in Local/Standalone mode, by making changes in the configuration files , we can convert it into the desired mode.

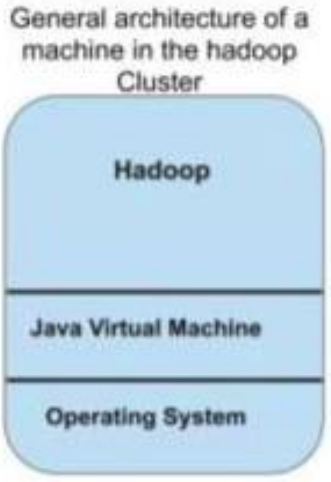


Figure 5. Hadoop Set up

Local or Standalone mode has all the daemons running on a single Java Virtual Machine. It is mainly used for debugging and developing purposes. It is not used for testing and production.It is set up and run on a single machine with a local File System.

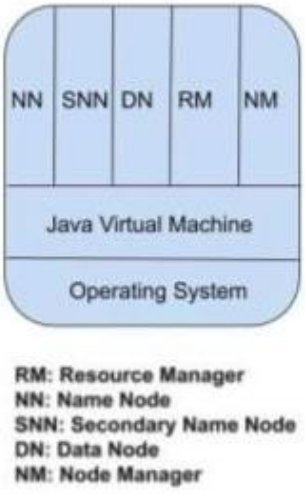


Figure 6. Hadoop Local Mode

Pseudo Distributed Mode of Hadoop has all its Daemons running on different individual JVMs.All the set up is done on a single machine.It is used for development and testing and the file system used here is HDFS . I have installed hadoop on my machine in pseudo distributed mode.

Figure 7. Hadoop Pseudo Distribution Mode

To set Hadoop in Pseudo Distributed Mode on Ubuntu I had to first download Java with the jre and the jdk then hadoop from the apache website. Then I had to untar the file and rename it to hadoop. Then I configured the environment variables so that I could access hadoop from anywhere. Then I modified the core-site.xml ,hdfs-site.xml, mapred-site.xml files. Then I formatted the namenode and started the namenodes,datanodes,secondary name nodes using start-dfs.sh, and resource manager, node manager using start-yarn.sh.HDFS comes with the hadoop installation.

Fully Distributed Mode of Hadoop has all its master daemons running on separate machines. Master daemons include NameNode, Secondary Name Node,Resource Manager. DataNode and NodeManger are kept on the same machine.The file system used here is HDFS.

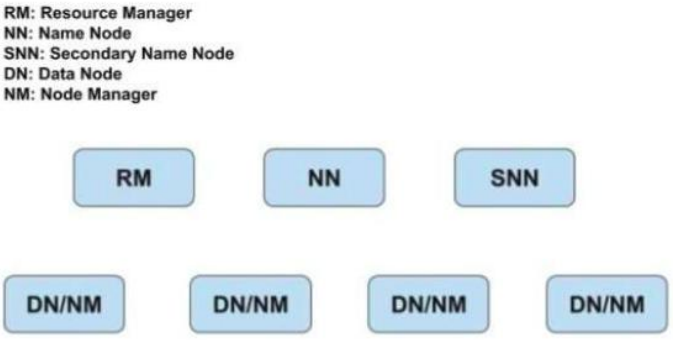


Figure 8. Hadoop Fully Distributed Mode

VII. HDFS

HDFS is a distributed file system in the hadoop architecture where all the files get stored in a Write Once Read Many(WORM) methodology. The file data is divided into blocks , by default the size of the blocks is 128MB but that is configurable. These blocks are stored on the datanodes and they are further replicated among the datanodes. By default the replication factor is 3 , but this is also configurable.

HDFS is conditioned to failures.When one data node is down due to hardware failure, the data can be accessed through any other data node where the data was replicated .HDFS is highly cost effective since it uses simple commodity hardware to process and store the data blocks. Commodity hardware is a piece of cheap hardware which has minimum specifications to qualify as a server.

Advantages of HDFS include that they can store large data like 100MBs and GBs . It follows the Write Once and Read Many times facility therefore it is useful for giving streaming data access facility.It needs a very low cost hardware.

Disadvantages of HDFS are that the data access speed is low therefore it cannot be used for fast data transaction. Small size of data also takes up a lot of space in the namenode as the namenode has the metadata stored.Also files cannot be written many times once they are stored in HDFS since the access time is low.

A.HDFS Architecture

HDFS consist of DataBlocks, NameNode,DataNode,Secondary NameNode,HDFS clientD

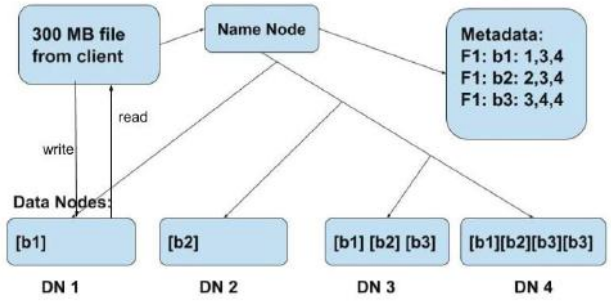


Figure 9.HDFS Architecture

a)Data Blocks

It is the minimum size of data stored in memory that can be read or written in one shot. Default size of a HDFS block is 128 MB but this can be changed by changing the property value in core-site.xml file . Data which is divided into different blocks is further stored in clusters.When the size of data is less than the size of the block , it is stored as a block but the whole block is not occupied here.In Figure 9. The file from the client is of 300MB , the size of the data block is default .Therefore the file will be divided into three blocks namely b1, b2, b3. b1 and b2 will be of size 128MB and b3 will be of size 128 MB but the occupied storage will be of 44MB. These blocks are stored in the clusters, here DN1, DN2, DN3, DN4.

b)Name Node

It is like the master node in Hadoop Master Slave architecture.It is like a controller or the manager of HDFS.It does not store any data but it holds the metadata of all the files as shown in the Figure9. File F1 has three blocks replicated 3 times and that metadata is stored in the namenode.Namenode is also responsible for storing data related to Data nodes and managing the file-system namespace .It controls access of different clients into the data blocks.It periodically checks for

the availability of the data nodes .The HDFS namespace is a hierarchy of files and directories. Files and directories are stored in the namenode by the inodes which record all the attributes of the files like the permissions , modification information, access times, namespace and disk space taken[1].

c)Data Node

These are the worker nodes , all the data blocks are stored in the data nodes. They send a report periodically to the namenode known as the Health Report.Default time is 3seconds. This report consists of different information including the data blocks like their location and the list of data blocks they are storing.Data nodes are generally commodity hardware so they are bound to fail ,but due to the replication of data the client accesses the data from other data nodes where the data is replicated.Once the block is stored in the data node then it replicates it among other data nodes.These data nodes carry out the tasks of creating , deleting data blocks as per the instruction of the namenode. DataNode identifies block replicas in its possession to the NameNode by sending the block report. This block report consists of the block id, generation stamp and length of each block replica[1].

d)Secondary Name Node

It is another specially dedicated node and it is used to take the checkpoints of the file system. Checkpoint of a file system gives the current state of the file system, it mainly consists of the file directory tree stored in the namenode which gives intel about all the data block locations.Secondary NameNode cannot substitute the NameNode but it helps the namenode retrieve information about the data blocks in case of failures.

e)HDFS Client

Users of HDFS are not located where the actual servers are placed. It is not possible to have users at the data center where the server is located, but a client methodology can be implemented to access data remotely. HDFS client is a code library that exports the HDFS file system interface.It also provides and Application Programming Interface which helps the user to locate file blocks and write MapReduce jobs accordingly to access the data faster.[1]

B.HDFS Reading and Writing

The two main operations in HDFS are reading and writing.

Reading the already stored file in HDFS . To read a file from HDFS the client sends a request for the metadata to the name node, as shown in Figure 10.

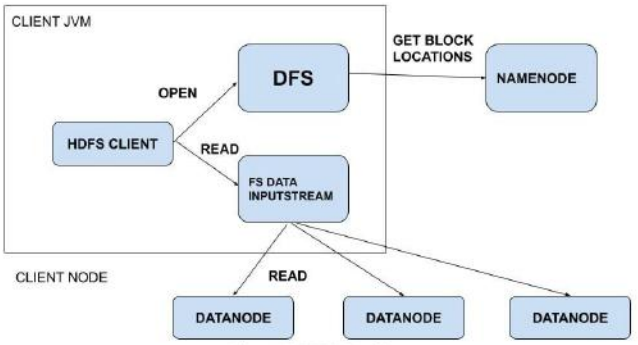


Figure 10.HDFS Reading

NameNode responds with a number of blocks ,location and information about replicas and other details.Client communicates with the DataNodes and reads the data in a parallel fashion. When the whole data is read, it combines the blocks as an original file [1].During the reading process in case a datanode fails then the client can access that block from another data node by taking the information about the location of that block from the metadata that it had received from the NameNode.

Writing the data in HDFS. To write a file into the HDFS the client sends a request to the namenode.NameNode responds and sends the metadata including the number of blocks ,locations of blocks already stored and other details.Client then splits the file(data) into blocks and starts sending them to the DataNodes.Once the block is received by the DataNode , it is replicated and stored into other datanodes.The procedure is shown in Figure 11.

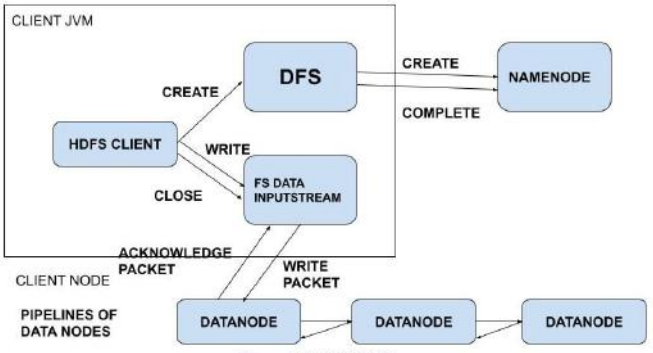
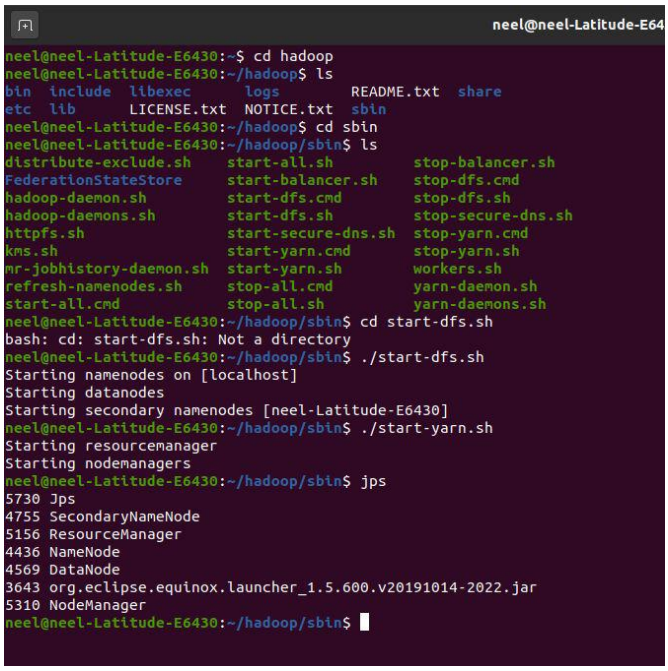


Figure 11.HDFS Writing

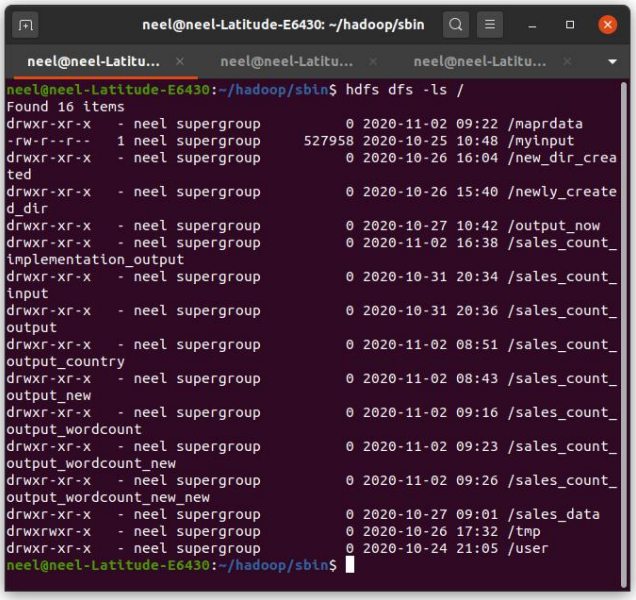
C.HDFS Common Commands

Like other file systems HDFS also provides the operational functionalities to read a file, write to a file, create new directories, and new files.The basic functionalities are displayed below.

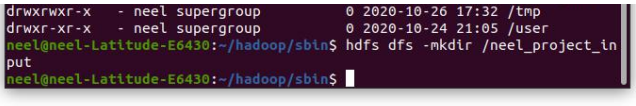
a)Starting the HDFS



b)List directories in HDFS
hdfs dfs -ls /

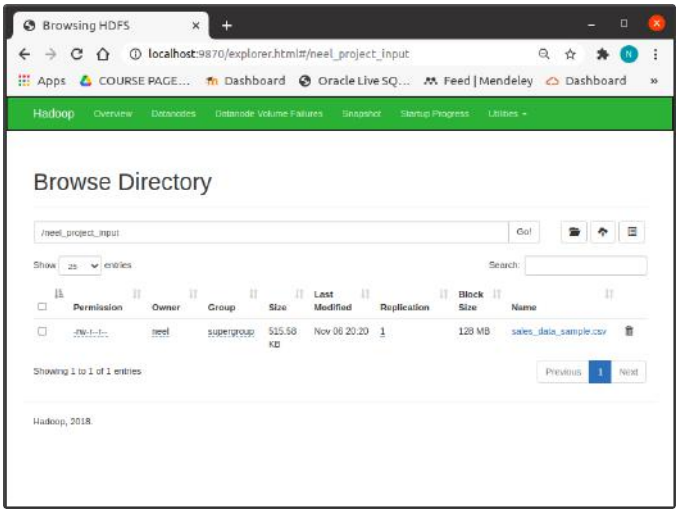


c)Make a new directory in HDFS
hdfs dfs -mkdir /neel_project_input

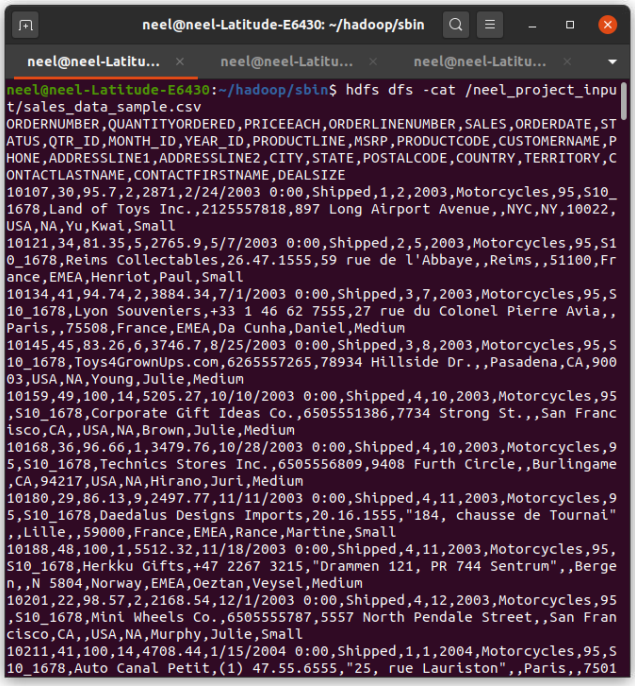


d)Remove a directory in HDFS
hdfs dfs -rmdir /forremoval

e)Copy a file from local disk to HDFS
hdfs dfs -put /home/neel/Downloads/sales_data_sample.csv /neel_project_input/

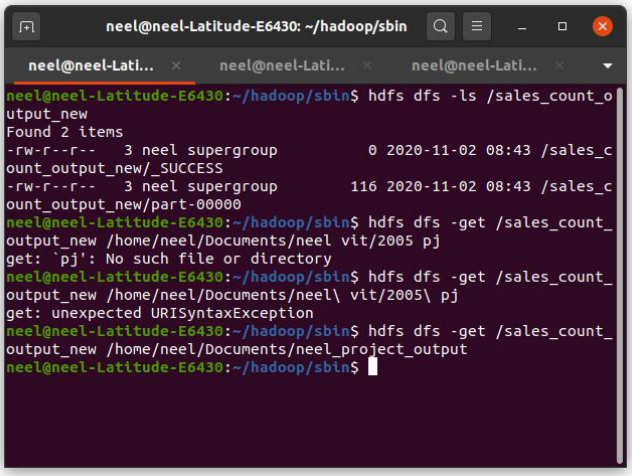


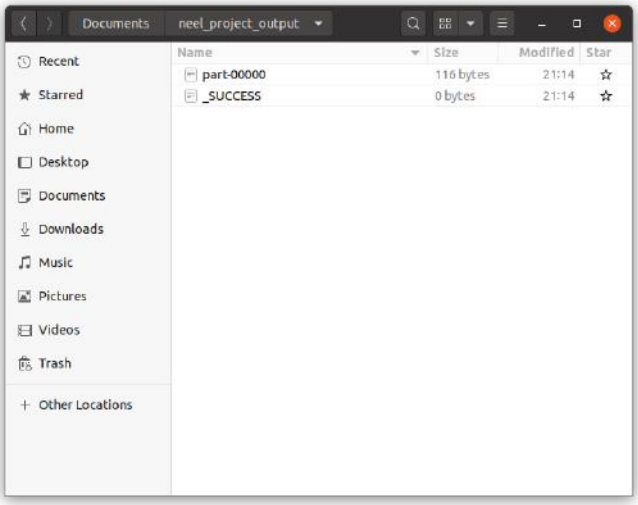
f)Display the contents of a file in HDFS
hdfs dfs -cat /neel_project_input/sales_data_sample.csv



g)Copy the file from HDFS to local disk

hdfs dfs -get /sales_count_output_new /home/neel/Documents/neel_project_output





VIII. HIVE

A.Introduction

It is a data warehousing framework built on top of hadoop.It is a tool to process structured data in Hadoop.It resides on top of Hadoop and helps in summarizing Big Data and makes querying and analysing easy. It enables the user to write SQL like HIVEQL queries which run MapReduce Jobs in the backend and fetch and analyse the data.It is not that good to build machine learning algorithms on it ,but data analysis can be easily done by this tool by grouping data.Initiated by facebook since it was collecting about 15TB of data per day and it had a total of 2PB of data, later taken up by Apache. Now Apache Hive is open source and also used by Amazon.

Features of Hive include storing schema in a database and processing data in HDFS. Online Transactional Processing can be done using HIVE but it is not the best way of doing OLTP.It provides SQL like query language called HIVEQL.It is quite familiar to most of the programmers and it is fast, scalable and extensible.

It is the best alternative for MapReduce jobs.Hive organizes data into tables therefore it becomes a means for attaching the structure to data stored in HDFS.Hive enables schema flexibility and evolution .Tables can be partitioned and bucketed. Apache Hive tables are directly defined in HDFS.JDBC/ODBC drivers are available to run Hive on Zeppelin which is a data visualization notebook[30].Hive also enables the user to do summarization of data, query the data .It is fast and scalable. File access in Hive can be done using various data stores like HBase and HDFS.

B.Comparison with traditional database

Traditional RDBMS cannot handle very large data since it is not scalable.Traditional database the data is checked before it is loaded into the database known as the schema on write.Apache Hive follows another methodology called as Schema on Read which means the data is not checked when it is stored into the database. [30]

Characteristics	RDBMS	HIVE
Methodology	Schema on Write: Checks the data before entry	Schema on Read: Does not check the data while it is stored
Speed	Not fast in loading	Fast loading
Querying	Provides fast querying	Slow querying
Type of data	Cannot store unstructured data	Can store semi-structured and structured data

Table 2. Comparison between RDBMS and HIVE

C. HIVE Architecture

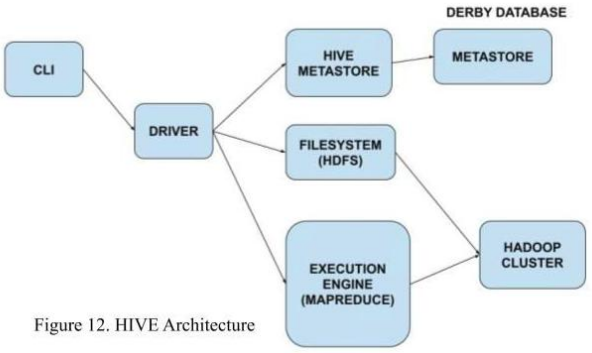


Figure 12. HIVE Architecture

HIVE mainly consists of Metastore,Driver,Compiler,Optimizer,Executer and the Shell.Metadata for each table is stored. The driver tracks the dataset in the cluster using this metadata. This metadata is stored in a RDBMS like MySQL.The Driver is the controller and it receives the HIVE QL statements .Once it receives the statements it creates a session , it then starts the monitoring cycle and progress of execution .It then stores the metadata. Finally it collects the data or query result. Compiler is the next important component of HIVE.It takes in the HIVE Query and creates an execution plan including tasks , it also creates steps

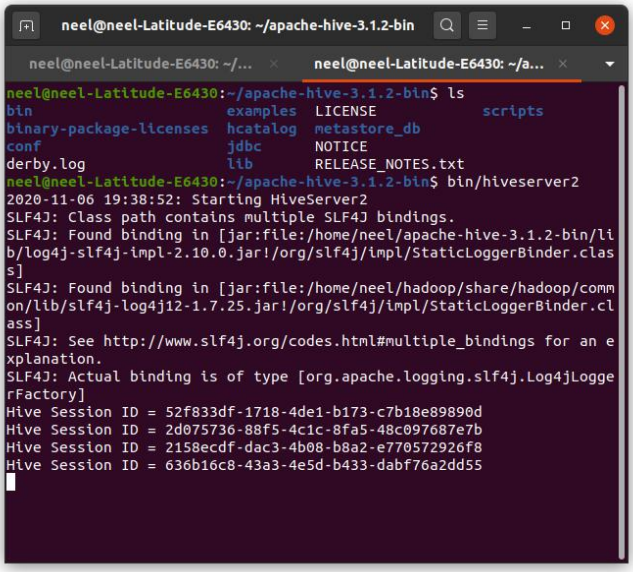
for MapReduce jobs.It creates a Directed Acyclic Graph which helps in conveying the plan ahead. Optimizer transforms this DAG into an optimized one and aggregates the transformations together.For instance it converts a pipeline of joins to a single join. It also splits the tasks like transforming the data before reducing operation.Executor takes in the tasks from the Optimizer and finally performs the Map and Reduce tasks.Hive Shell is used to interact with Hive in a Command Line Interface Format.

D.Basic HIVE Functionalities

HDFS is already started.Now to run HIVE, I have used beeline command line interface to connect to the driver and I have used hiveserver2 to run HIVE.

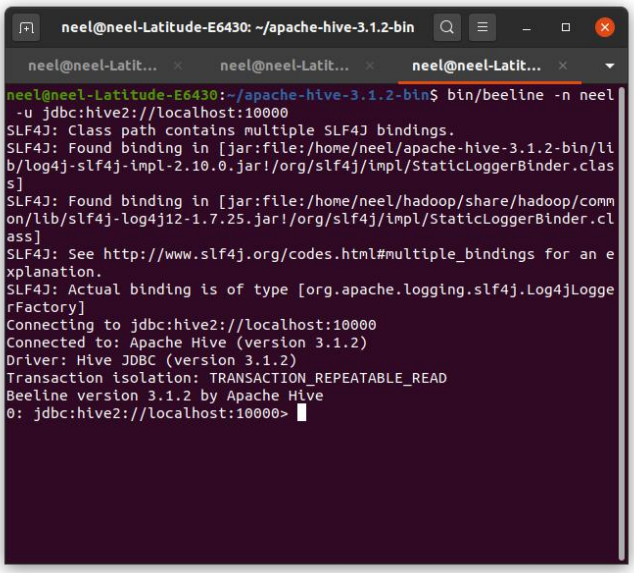
1)Starting hiveserver2

$HIVE_HOME/bin/hiveserver2



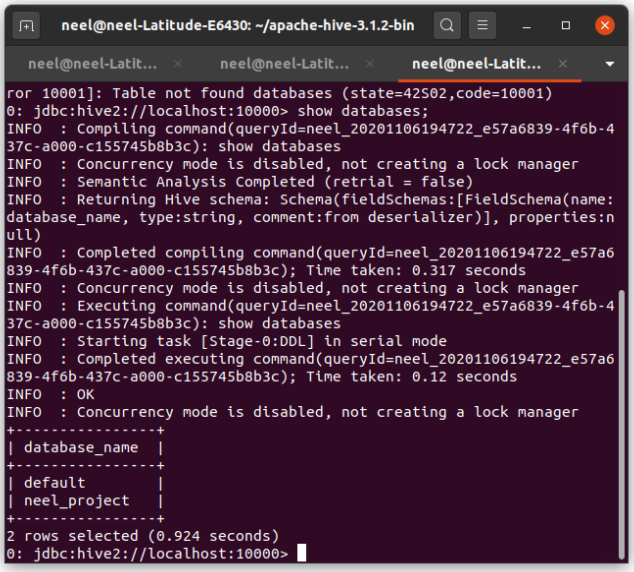
2)Connecting hdfs with hive server using jdbc connection:

$HIVE_HOME/bin/beeline -n neel -u jdbc:hive2://localhost:10000



3)Displaying the databases present in hdfs:

show databases;

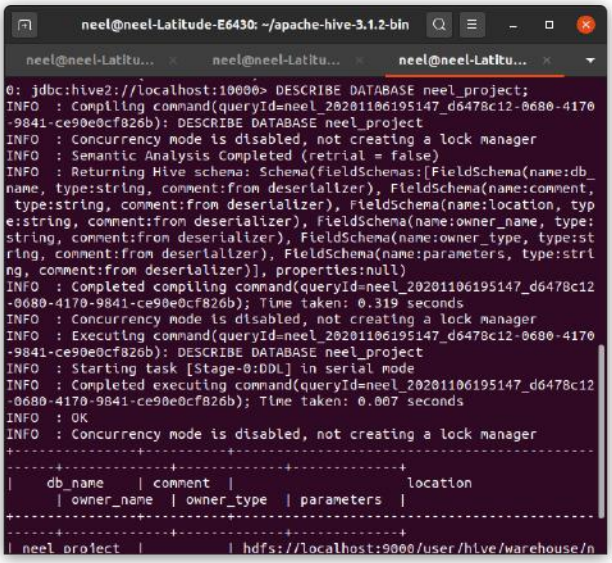


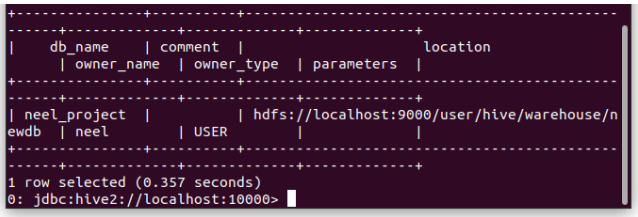
4)Creating a database:

CREATE DATABASE IF NOT EXISTS neel_project LOCATION ‘/user/hive/warehouse/newdb’

5)Describing the created database:

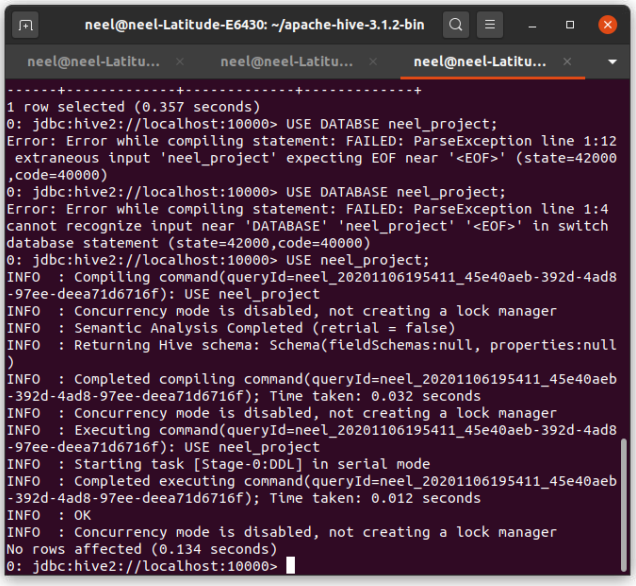
DESCRIBE DATABASE neel_project;





6)Using the database:

USE neel_project;



7)Creating the table for importing data from hdfs:

CREATE TABLE IF NOT EXISTS project_sales_data_new(Ord_no integer, Qty_ord integer, Price_perunit double , Orderline_number integer, sales double, date_order date,status string ,qtr_id integer, month_id integer, year_id integer, product_line string, msrp integer, product_code string , cust_name string, phone string, add_line1 string, add_line2 string , city string, state string , postal_code string , country string , territory string, last_name string , first_name string, deal_size string)

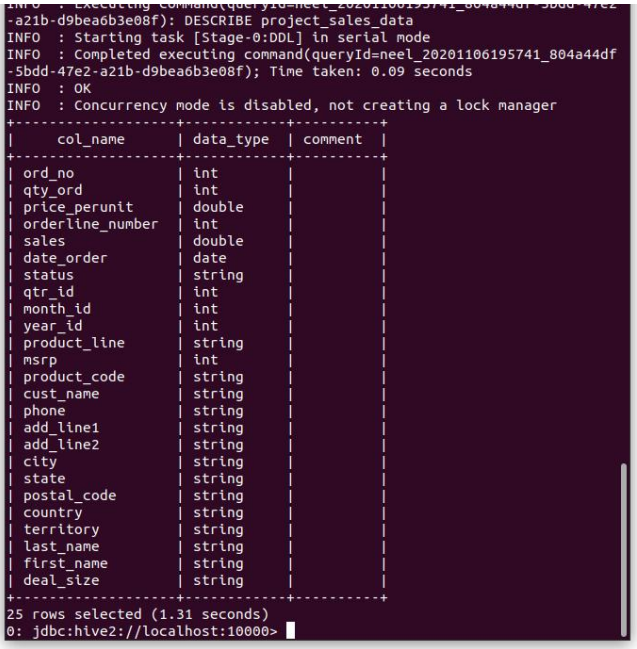
ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

8)Metadata of the table project_sales_data:

DESCRIBE project_sales_data;



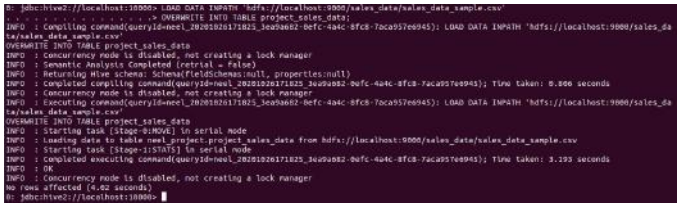
9)Loading the data into hdfs from local disk:

hdfs dfs -put /home/neel/Downloads/sales_data_sample.csv /neel_project_input/

10)Loading the data in hdfs into hive table :

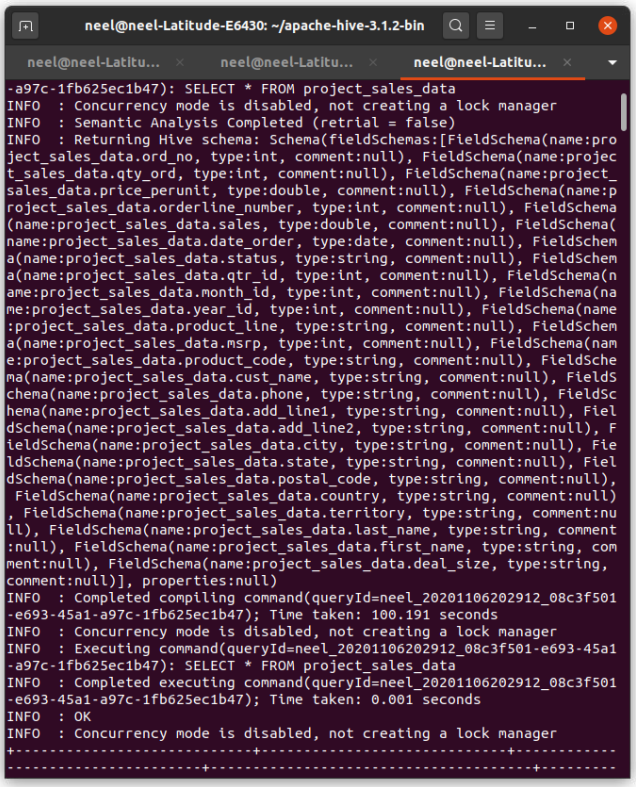
Sales_data_sample.csv is the file that we are going to load into a hive table:

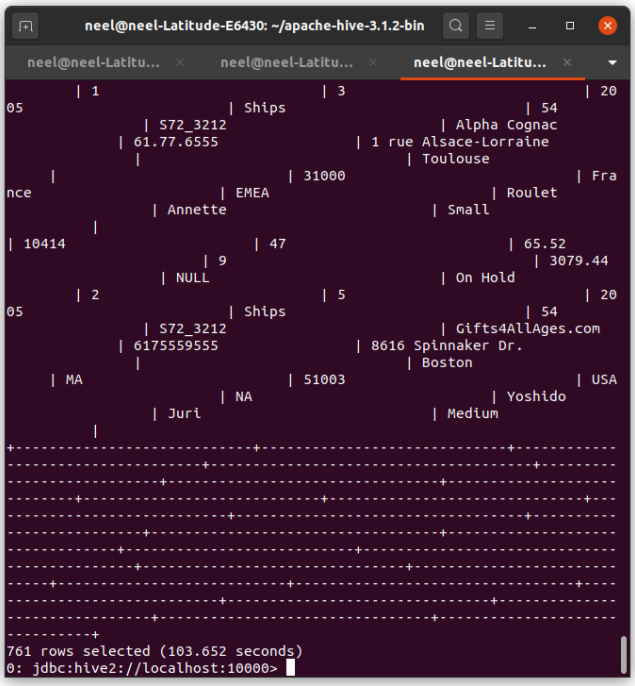
LOAD DATA INPATH hdfs://localhost:9870/neel_project_input/sales_data_sample.csv



11)Verifying that the data has loaded into the hive table:

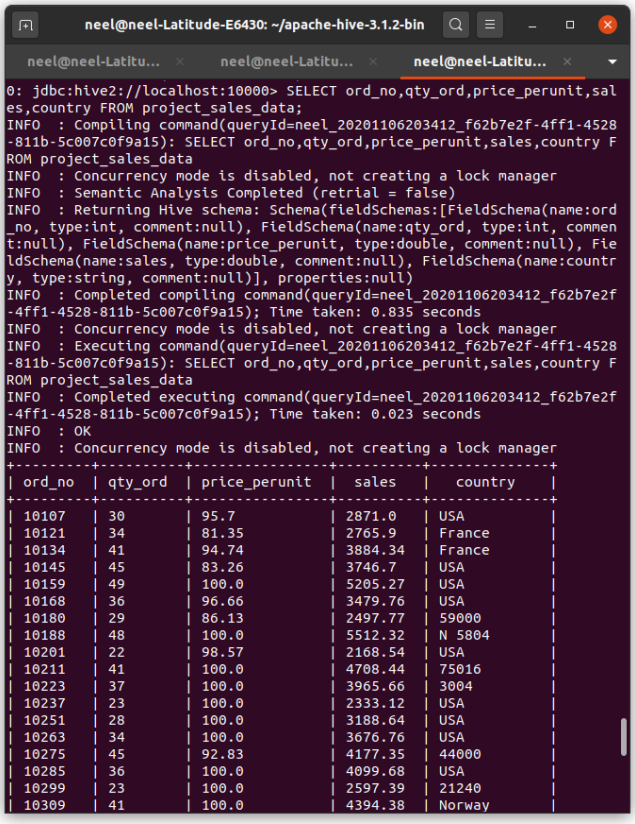
SELECT * FROM project_sales_data;

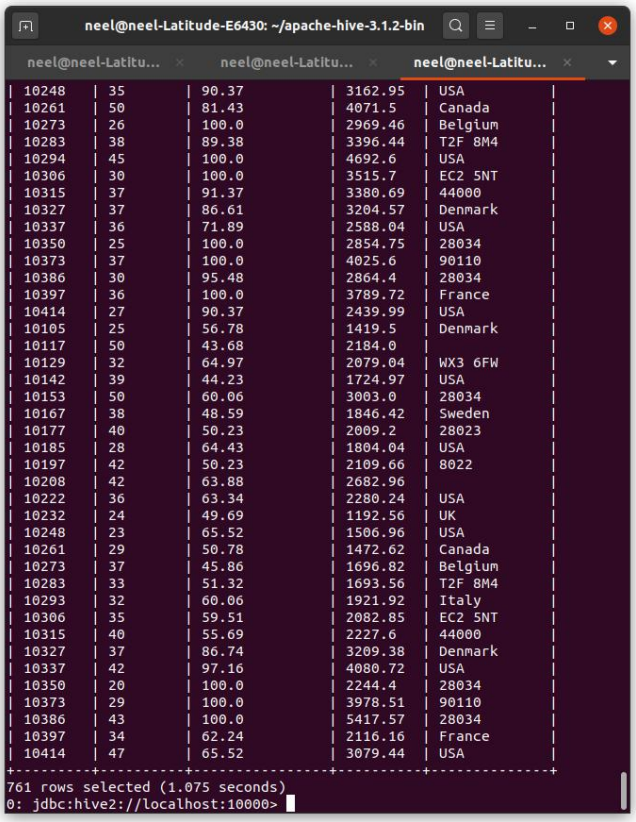




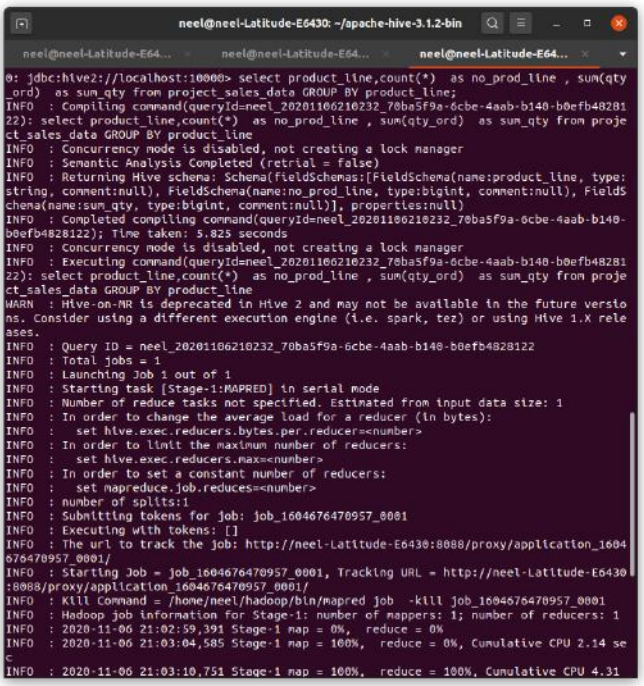
12)Displaying the table where the data is imported:

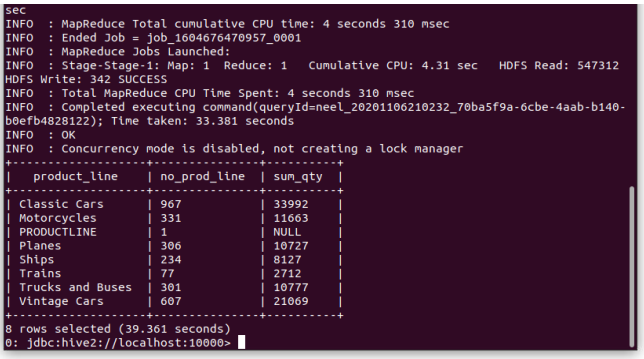
SELECT ord_no,qty_ord,price_perunit,sales,country FROM project_sales_data;





E.Sales Data Analysis using HiveQL
Dataset taken from kaggle[37]
1)counting the total quantity sold and the number of products per product_line
select product_line,count(*) as no_prod_line , sum(qty_ord) as sum_qty from project_sales_data GROUP BY product_line;





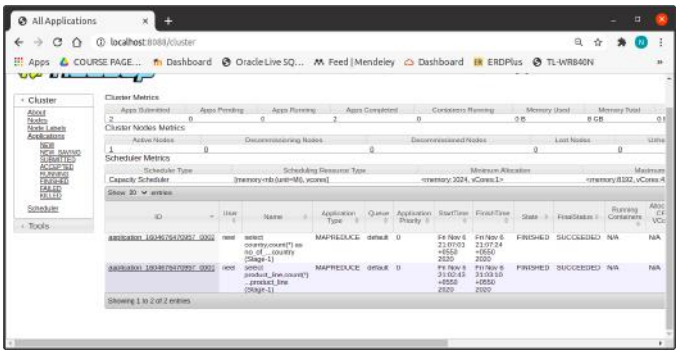
2)counting the total quantity sold and the number of products per country:
select country,count(*) as no_of_sales_percountry , sum(qty_ord) as sum_qty from project_sales_data GROUP BY country;

```
neel@neel-Latitude-E6430: ~/apache-hive-3.1.2-bin
0: jdbc:hive2://localhost:10000> select country,count(*)  as no_of_sales_percountry , sum(qty_ord)  as sum_qty from project_sales_data GROUP BY country;
INFO  : Compiling command(queryId=neel_20201106210658_116605ec-981f-4433-8f8f-da20c33e5f61): select country,count(*)  as no_of_sales_percountry , sum(qty_ord)  as sum_qty from project_sales_data GROUP BY country
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:country, type:string, comment:null), FieldSchema(name:no_of_sales_percountry, type:bigint, comment:null), FieldSchema(name:sum_qty, type:bigint, comment:null)], properties:null)
INFO  : Completed compiling command(queryId=neel_20201106210658_116605ec-981f-4433-8f8f-da20c33e5f61); Time taken: 0.516 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=neel_20201106210658_116605ec-981f-4433-8f8f-da20c33e5f61): select country,count(*)  as no_of_sales_percountry , sum(qty_ord)  as sum_qty from project_sales_data GROUP BY country
WARN  : Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
INFO  : Query ID = neel_20201106210658_116605ec-981f-4433-8f8f-da20c33e5f61
INFO  : Total jobs = 1
INFO  : Launching Job 1 out of 1
INFO  : Starting task [Stage-1:MAPRED] in serial mode
INFO  : Number of reduce tasks not specified. Estimated from input data size: 1
INFO  : In order to change the average load for a reducer (in bytes):
INFO  :   set hive.exec.reducers.bytes.per.reducer=<number>
INFO  : In order to limit the maximum number of reducers:
INFO  :   set hive.exec.reducers.max=<number>
INFO  : In order to set a constant number of reducers:
INFO  :   set mapreduce.job.reduces=<number>
INFO  : number of splits:1
INFO  : Submitting tokens for job: job_1604676470957_0002
INFO  : Executing with tokens: []
INFO  : The url to track the job: http://neel-Latitude-E6430:8088/proxy/application_1604676470957_0002/
INFO  : Starting Job = job_1604676470957_0002, Tracking URL = http://neel-Latitude-E6430:8088/proxy/application_1604676470957_0002/
INFO  : Kill Command = /home/neel/hadoop/bin/mapred job  -kill job_1604676470957_0002
INFO  : Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
INFO  : 2020-11-06 21:07:13,861 Stage-1 map = 0%,  reduce = 0%
INFO  : 2020-11-06 21:07:18,996 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.05 sec
INFO  : 2020-11-06 21:07:24.118 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.22
```

```
neel@neel-Latitude-E6430: ~/apache-hive-3.1.2-bin
sec
INFO  : MapReduce Total cumulative CPU time: 4 seconds 220 msec
INFO  : Ended Job = job_1604676470957_0002
INFO  : MapReduce Jobs Launched:
INFO  : Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.22 sec   HDFS Read: 547441 HDFS Write: 1310 SUCCESS
INFO  : Total MapReduce CPU Time Spent: 4 seconds 220 msec
INFO  : Completed executing command(queryId=neel_20201106210658_116605ec-981f-4433-8f8f-da20c33e5f61); Time taken: 27.441 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
+--------------+--------------------------+----------+
|   country    | no_of_sales_percountry   | sum_qty  |
+--------------+--------------------------+----------+
|              | 192                      | 6827     |
| 106-0032     | 32                       | 1150     |
| 1203         | 31                       | 1078     |
| 13008        | 25                       | 804      |
| 2            | 16                       | 490      |
| 2068         | 46                       | 1469     |
| 2067         | 46                       | 1601     |
| 21240        | 30                       | 1051     |
| 28023        | 13                       | 468      |
| 28034        | 259                      | 9327     |
| 3004         | 55                       | 1926     |
| 3150         | 23                       | 765      |
| 4101         | 15                       | 545      |
| 44000        | 60                       | 2102     |
| 50739        | 26                       | 936      |
| 59000        | 20                       | 699      |
| 60528        | 22                       | 811      |
| 67000        | 19                       | 779      |
| 71270        | 21                       | 668      |
| 75016        | 27                       | 1001     |
| 78000        | 18                       | 637      |
| 8022         | 23                       | 882      |
| 80686        | 14                       | 401      |
| 90110        | 32                       | 1110     |
| Austria      | 55                       | 1974     |
| B-6000       | 8                        | 278      |
| Belgium      | 25                       | 796      |
| COUNTRY      | 1                        | NULL     |
| Canada       | 44                       | 1420     |
```

```
| Denmark      | 63                       | 2197     |
| EC2 5NT      | 51                       | 1778     |
| FIN-02271    | 30                       | 1031     |
| France       | 81                       | 2804     |
| Italy        | 113                      | 3773     |
| N 5804       | 29                       | 973      |
| Norway       | 32                       | 1082     |
| Osaka        | 20                       | 692      |
| Philippines  | 26                       | 961      |
| S-958 22     | 19                       | 647      |
| Singapore    | 36                       | 1236     |
| Sweden       | 38                       | 1359     |
| T2F 8M4      | 26                       | 873      |
| UK           | 26                       | 895      |
| USA          | 969                      | 34491    |
| WA1 1DP      | 12                       | 357      |
| WX1 6LT      | 29                       | 1046     |
| WX3 6FW      | 26                       | 937      |
+--------------+--------------------------+----------+
47 rows selected (28.007 seconds)
0: jdbc:hive2://localhost:10000>
```

On localhost:8088: log of the task executed:



IX. MAP REDUCE

A.Introduction

It is one of the most important components of the Hadoop ecosystem.It is designed to process a large amount of data in parallel by dividing the work into some smaller and independent tasks.Whole job is taken from user and is further divided into smaller independent tasks and assigned to worker nodes.Map Reduce programs take input as a list and also return a list as output.[30]

B.Map,Reduce and MapReduce

The Map function is the method defined inside the Mapper Class . It takes in a set of key value pairs. The input can be in a structured format or in an unstructured form.Keys reference to input files and values is the dataset.User can create the business logic according to his needs.The task is applied to every input value.

Reducer function is the method defined inside the Reducer Class. It takes in the key value pairs which were outputted by the Mapper Class.User defined functions can be written in the reduce method.

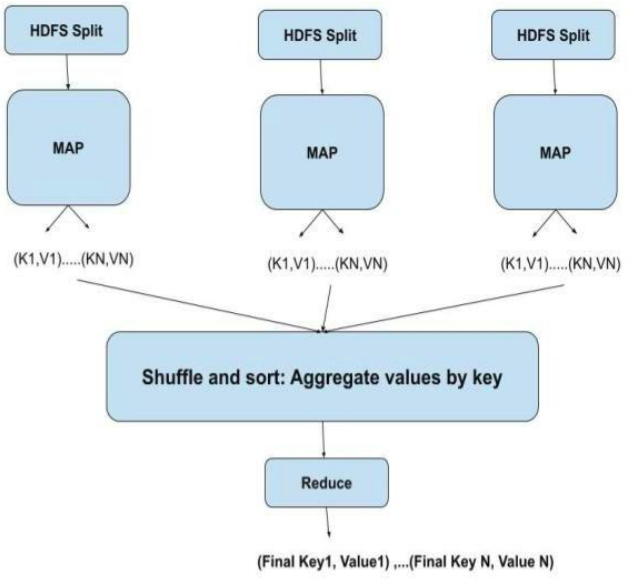


Figure 13. MapReduce Process

C.Shuffling and Sorting

Shuffling is the first functionality that is done after the Mapper class gives its output. Mapper creates the intermediate key-value pairs and transfers them to the reducer task.Using the shuffling procedure the, the system acn sort the data using key values[30].Shuffling task begins after a certain mapping tasks are done, it does not wait for the mapper class to finish its tasks.

Sorting takes place after Shuffling is over.MapReduce framework automatically sorts data on key values on the output of Mapper class .The framework makes sure that all the key-value pairs are sorted before they are passed into the Reducer class so that the reducer class can easily identify the key-value pairs and perform its functionality.If no reducer task is made by the user then there is no shuffling or sorting.

D.Job Scheduling

Clients or the users creates different types of jobs to perform data analysis .Jobs includes all kinds of functionalities like writing to a HDFS directory or reading from an HDFS directory, calling the mapper class , or the reducer class.There are three types of job scheduling algorithms which the map reduce framework performs. First In First Out (FIFO), it is a default job scheduling algorithm performed by the Job Tracker. Capacity Scheduler , this is a default job scheduling method performed by the resource manager.Fair Scheduler is the third type of job scheduler.

E.Maven Java API and its interaction with HDFS

Maven is used for Java based projects . It is a tool that helps us to download various dependencies for our code.It creates the .jar files for the project which will be deployed on HDFS to perform MapReduce functionality natively on the command line of HDFS.Maven is also developed by the Apache group.Maven is based on the Project Object Model(POM) and focuses on simplification and standardization of the building process.POM has a POM.xml file where we can list the dependencies in the form of properties and values . Here I have installed all the hadoop and mapreduce related dependencies by mentioning the version of hadoop and the name of the packages.Using this XML file Maven downloads all the dependencies and enables us to use them.

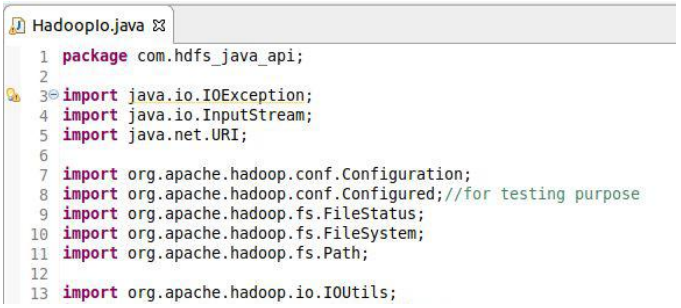
Following code shows the interaction of Maven API with HDFS.

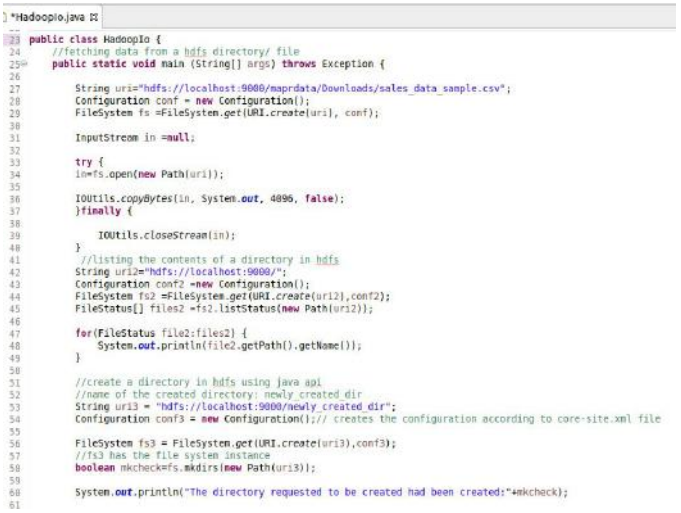
Code:

Setup for HDFS:

Uploading a file in the local directory to hdfs using command line:

hdfs dfs -put /home/neel/Downloads/sales_data_sample_new.csv /sales_data

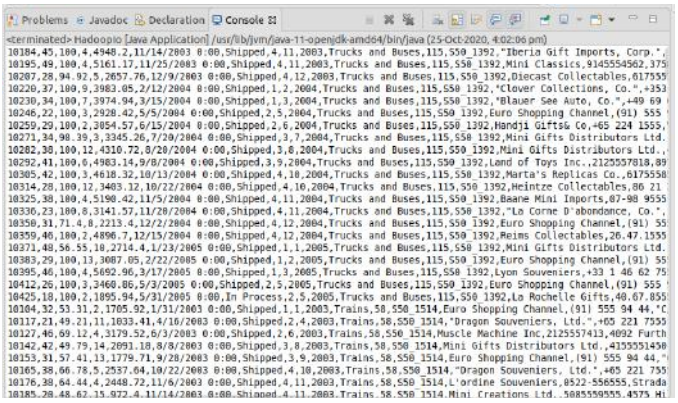




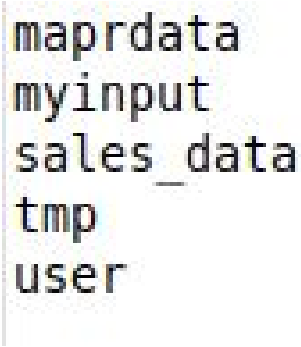
```java
*Hadooplo.java
63          //uploading a file to hdfs
64
65          //fielpath for the file in the local file system:
66          String localPath="/home/neel/Documents/neel vit/2005 pj/19BCE0990.doc";
67
68
69          String uri4 = "hdfs://localhost:9000/";
70
71          //hdfs directory : where the file will be stored:
72          String dir_hdfs="hdfs://localhost:9000/new_dir_created";
73
74          Configuration conf4=new Configuration();
75          FileSystem fs4= FileSystem.get(URI.create(uri4),conf4);
76
77          fs4.copyFromLocalFile(new Path(localPath),new Path(dir_hdfs));
78
79          //downloading from hdfs :
80          //fielpath for the file in the local file system:
81          String localPath2="/home/neel/Documents/neel vit/2005 pj/downloads_project";
82
83
84          String uri5 = "hdfs://localhost:9000/";
85
86          //hdfs directory : where the file will be stored:
87          String dir_hdfs2="hdfs://localhost:9000/new_dir_created/19BCE0990.doc";
88
89          Configuration conf5=new Configuration();
90          FileSystem fs5= FileSystem.get(URI.create(uri5),conf5);
91
92          fs5.copyToLocalFile(new Path(dir_hdfs2),new Path(localPath2));
93
94          //deleting a file in hdfs
95
96          String uri6="hdfs://localhost:9000/new_dir_created/19BCE0990.doc";
97          Configuration conf6=new Configuration();
98          FileSystem fs6=FileSystem.get(URI.create(uri6),conf6);
99          fs6.delete(new Path(uri6),true);
100         }
101 }
```

Outputs:

1)import file from hdfs using Java API:



2)List the directories in hdfs

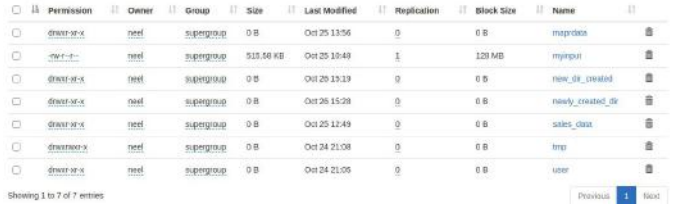


3)creation of a directory:

```
The directory requested to be created had been created:true
```

Before creation:

After creation:



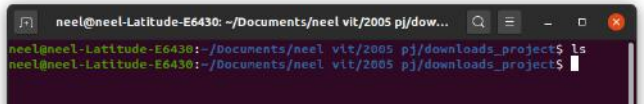
4)Upload file to hdfs using java api

Before uploading:

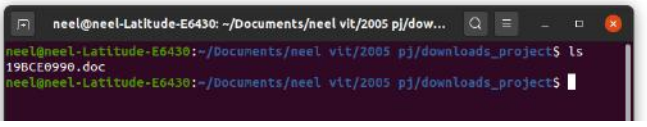
After uploading:

5)downloading file from hdfs using java api

Before downloading:



After downloading:



6)delete file in hdfs :

Before delete:

After delete:

F.Sales data analysis Algorithm

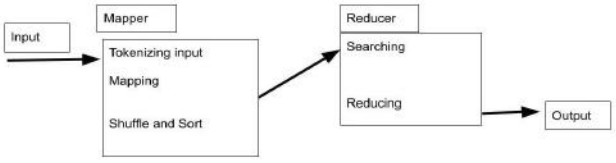


Figure 14. MapReduce Algorithm

Input:
Sales_data.csv

Input: sales_data_sample.csv

```
neel@neel-Latitude-E6430:~/hadoop/etc/hadoop$ hdfs dfs -cat /sales_count_Input/sales_da
ta_sample.csv
ORDERNUMBER,QUANTITYORDERED,PRICEEACH,ORDERLINENUMBER,SALES,ORDERDATE,STATUS,QTR_ID,MON
TH_ID,YEAR_ID,PRODUCTLINE,MSRP,PRODUCTCODE,CUSTOMERNAME,PHONE,ADDRESSLINE1,ADDRESSLINE2
,CITY,STATE,POSTALCODE,COUNTRY,TERRITORY,CONTACTLASTNAME,CONTACTFIRSTNAME,DEALSIZE
```

PRODUCTLINE [10]
COUNTRY[20]

Figure 15.Input for Sales Data

MAP

FOR PRODUCTLINE

(PRODUCTLINE,1)
(Motorcycles,1)
(Classic Cars,1)
(Motorcycles,1)
(Classic Cars,1)
.
.
.
(Planes,1)
(Motorcycles,1)
(Ships,1)
.
.
.
(Trains, 1)
.
(Trucks and Buses,1)
(Vintage Cars,1)
.
.
.

FOR COUNTRY

(COUNTRY,1)
(USA,1)
(France,1)
(USA,1)
(Norway,1)
.
.
.
(Austria,1)
(Australia,1)
(Sweden,1)
.
.
.
(Spain, 1)
.
(USA,1)
(Sweden,1)
.
.
.

Figure 16. Map method for Sales Data

SHUFFLE AND SORT

FOR PRODUCTLINE

(Classic Cars,[1,..])
(Motorcycles,[1,..])
(PRODUCTLINE,[1,..])
(Planes,[1,..])
(Ships,[1,..])
(Trains,[1,..])
(Trucks and Buses,[1,..])
(Vintage Cars,[1,..])

FOR COUNTRY

(Austria,[1,..])
(B-6000,[1,..])
(Belgium,[1,..])
(COUNTRY,[1,..])
(Canada,[1,..])
(Denmark,[1,..])
(EC2 5NT,[1,..])
(FIN-0227130,[1,..])
(France,[1,..])
(Italy 113,[1,..])
(N 5804,[1,..])
(Norway,[1,..])
(Osaka,[1,..])
(Philippines,[1,..])
(S-958 22,[1,..])
(Singapore,[1,..])
(Sweden,[1,..])
(T2F 8M4,[1,..])
(UK,[1,..])
(USA,[1,..])

Figure 17. Shuffle and Sort for Sales Data

REDUCE

FOR PRODUCTLINE

(Classic Cars,1)
(Motorcycles,1)
(PRODUCTLINE,1)
(Planes,1)
(Ships,1)
(Trains,1)
(Trucks and Buses,1)
(Vintage Cars,1)

FOR COUNTRY

(Austria,1)
(B-6000,1)
(Belgium,1)
(COUNTRY,1)
(Canada,1)
(Denmark,1)
(EC2 5NT,1)
(FIN-0227130,1)
(France,1)
(Italy 113,1)
(N 5804,1)
(Norway,1)
(Osaka,1)
(Philippines,1)
(S-958 22,1)
(Singapore,1)
(Sweden,1)
(T2F 8M4,1)
(UK,1)
(USA,1)

Figure 18.Reduce method for Sales Data

OUTPUT

FOR PRODUCTLINE

Classic Cars 967
Motorcycles 331
PRODUCTLINE 1
Planes 306
Ships 234
Trains 77
Trucks and Buses 301
Vintage Cars 607

FOR COUNTRY

Austria 55
B-6000 8
Belgium 25
COUNTRY 1
Canada 44
Denmark 63
EC2 5NT 51
FIN-02271 30
France 81
Italy 113
N 5804 29
Norway 32
Osaka 20
Philippines 26
S-958 22 19
Singapore 36
Sweden 38
T2F 8M4 26
UK 26
USA 969

Figure 19. Output for Sales Data

G.Sales data analysis using MapReduce
Dataset taken from kaggle[37].
Code:
Driver Code:

*SalesCounterDriver.java

```
package sales_count;
//package name is sales_count here and this mapreduce program calculates the sales done according to
//country and according to Productline.

//output of the program will directly go into a directory named sales_count

//importing the necessary libraries to perform a job in hadoop mapreduce:
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;
//driver function which creates the job and calls the mapper and the reducer classes:
public class SalesCounterDriver {
    //this driver class is responsible for setting our MapReduce job to run in Hadoop.

    public static void main (String[] args ) {
        //arg[0] and arg[1] command line arguments are passed by the user in the command prompt
        //here the arg[0] will have the location of the input file in hdfs
        //and arg[1] is will have the location of directory in the hdfs where the output will be saved
        JobClient my_client = new JobClient(); //creating a job in mapreduce

        JobConf  job_conf = new JobConf(SalesCounterDriver.class);// creating the configuration of the job according to the
                                                                  // configurations mentioned in pom.xml:
                                                                  //these configurations include the hadoop and
                                                                  //mapreduce versions

        job_conf.setJobName("CountrySales");//providing a name for the job

        job_conf.setOutputKeyClass(Text.class);//the output class for the key will be of Text type:
                                               //Text type in hadoop library is similar to String in Java

        job_conf.setOutputValueClass(IntWritable.class);//the output class for the value will be of IntWritable type:
                                                        // IntWritable class is similar to Integer in Java

        job_conf.setMapperClass(sales_count.SalesCounterMapper.class); //Specifying and calling the Mapper function

        job_conf.setReducerClass(sales_count.SalesCounterReducer.class);//Specifying and calling the Reducer function

        job_conf.setInputFormat(TextInputFormat.class);//data type is set for the Input directory from where the data
                                                       //is imported(in hdfs)
        job_conf.setOutputFormat(TextOutputFormat.class);//data type is set for the Output directory where the output
                                                         //will be stored(in hdfs)


        FileInputFormat.setInputPaths(job_conf,new Path(args[0]));//taking in the argument arg[0] has the input path
        FileOutputFormat.setOutputPath(job_conf,new Path(args[1]));//taking in the argument arg[1] has the output path

        my_client.setConf(job_conf);
        try {//trying the job_conf first and returning any errors if any were found
            JobClient.runJob(job_conf);
        }catch (Exception e) {
            e.printStackTrace();

        }
    }
}
```

Mapper Code:

SalesCounterMapper.java

```
package sales_count;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class SalesCounterMapper extends MapReduceBase
implements Mapper<LongWritable, Text, Text, IntWritable>
{   //SalesCounerMapper function has to inherit the functions
    //and methods of the default: MapReduceBase class
    //also it is going to implent the functionality of the mapper
    //which takes in 4 arguments by defualt:
    // INPUTKEY , INPUTVALUE, OUTPUTKEY , OUTPUTVALUE
    //here:
    //INPUTKEY: Is of Type LongWritable(similar to Long datatype in Java)
    //INPUTVALUE: Is of type Text(similar to Sting in Java)
    //OUTPUTKEY: Is of Type Text( similar to string in Java)
    //OUTPUTVALUE: Is of type IntWritable(similar to Integer class in Java)
    private final static IntWritable one = new IntWritable(1);
    // new instance created for the IntWritable class
    public void map(LongWritable key, Text value, OutputCollector <Text, IntWritable>
            output, Reporter reporter) throws IOException{
        //map function that takes in the 4 arguments:
        // key: The input key is of type LongWritable in hadoop is similar
        //to Long in Java

        //value : The input value is of type Text, similar to String in java
        //output : The output is of type (key,value) where key: type is Text
        //and value: type is IntWritable
        //reporter: is of Class Reporter
        String valueString = value.toString();//input value converted to String
        //data type in Java from Text type in Hadoop
        String[] SingleCountryData = valueString.split(",");
        // since the input data is in the form of comma separated values(csv)
        //file, here the delimiter mentioned is a comma(",")
        //counting the number of sales done according to Productline
        output.collect(new Text(SingleCountryData[10]),one);
        //this is for counting the number of sales done according to country
        //output.collect(new Text(SingleCountryData[20]),one);
        //
    }
}
```

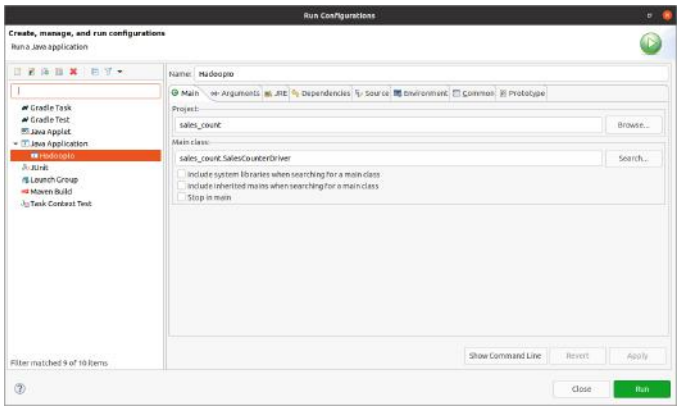
Reducer Code:

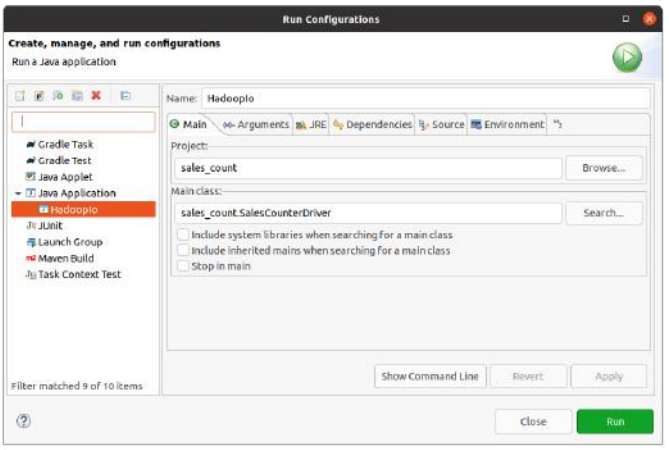
SalesCounterReducer.java

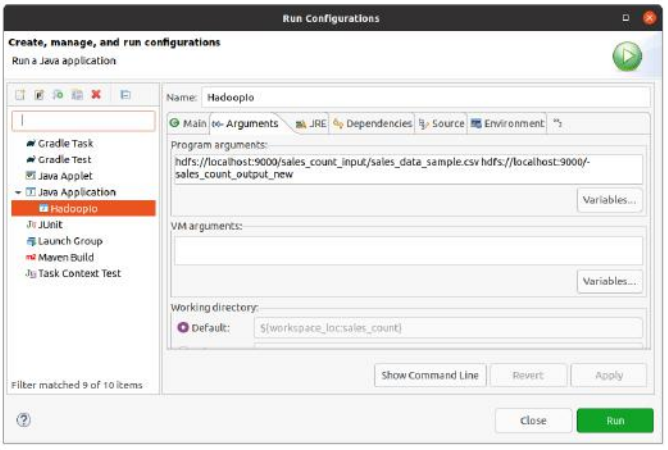
```
package sales_count;
import java.io.IOException;
import java.util.Iterator;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public  class SalesCounterReducer extends MapReduceBase
implements Reducer<Text, IntWritable,Text,IntWritable>  {
    //Reducer class extended by the functions and methods of MapReduceBase
    //class which is the default reducer class
    //also the reducer class is implementing the functionality of a
    //reducer and taking in 4 default arguments:
    //inputkey: Text type same as String in Java
    //inputValue: IntWritable type same as Integer in Java
    //outputKey: Text type same as String in Java(here ProductLine for
    //counting according to product line
    //and Country for counting according to country)
    //outputValue: IntWritable type which is similar to Integer class
    //in Java . This give out the frequency of
    //the input keys occuring

    //takes in the output key,value pairs from the mapper class
    //and gives out the keys and number of times these keys has occured
    //here when we are counting according to ProductLines : the keys are
    //the unique values of the column ProductLine
    //when we are counting according to Country: the keys are the unique
    //values of the countries

    public void reduce(Text t_key , Iterator<IntWritable> values,
            OutputCollector <Text,IntWritable> output , Reporter report)
            throws IOException{
        //reduce method:
        //t_key: is the key(type text) that is taken in from the
        //output of mapper class
        //values : are the list of values for each key therefore they
        //are of type Iterator which is also defined in hadoop libraries
        //output is of the type (key,value): (Text,IntWritable) :
        //here key is the Country for counting according to country and
        // is the Product Line for counting according to Productline
        //value is the frequency(number of times it has occured)
        //of Country or Product Line
        //Reporter is defined in the hadoop libraries to report
        //any errors during the process
        Text key =t_key;//key is output key of Mapper
        int frequency=0;//setting frequency to 0

        while(values.hasNext()) {//while the values (list) has a next
            //Element this loop will continue and increment the frequency counter
                                //in every iteration
            IntWritable value = (IntWritable) values.next();//value is
            //fetched in the value variable : here the value is 1 always
            //since every key(Country or ProductLine) is mapped to value 1.
            frequency+=value.get();//value.get() fetches value 1 as
            //Integer class in Java since it was fetched as an IntWritable class which
                                // is defined in the hadoop libraries.

        }
        output.collect(key, new IntWritable(frequency));
        //finally the output is displayed in the part-00000 file
        //format of the output will be:
        // key: name of country(first implementation) or Product line(second implementation)
        //frequency: number of times that key has occured
        // here the frequency was of Integer type so a new instance
        //of IntWritable class is made since the output
        // is in the hadoop system
    }

}
```

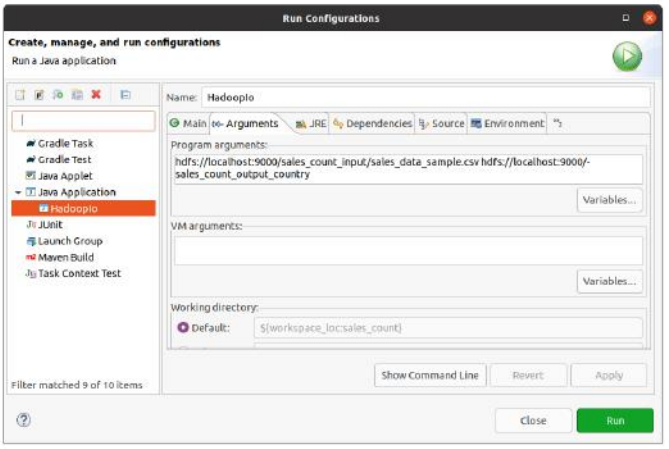
Inputs and Outputs:

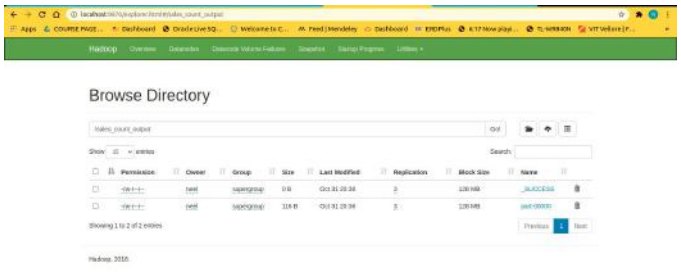
According to Product line:

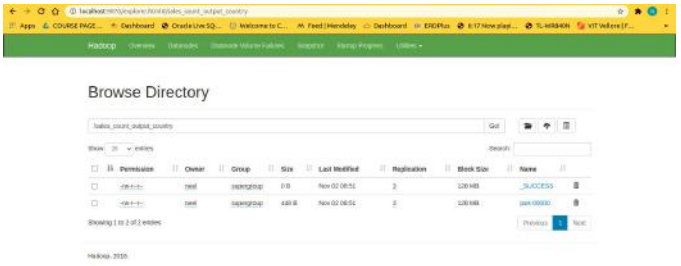













```
neel@neel-Latitude-E6430:~/hadoop/sbin$ hdfs dfs -cat /sales_count_output/part-
00000
Classic Cars    967
Motorcycles     331
PRODUCTLINE     1
Planes  306
Ships   234
Trains  77
Trucks and Buses        301
Vintage Cars    607
```

According to Country:

```
neel@neel-Latitude-E6430:~/hadoop/sbin$ hdfs dfs -cat /sales_count_output_country/part-00000
        192
106-0032        32
1203    31
13008   25
2       16
2060    46
2067    46
21240   30
28023   13
28034   259
3004    55
3150    23
4101    15
44000   60
50739   26
59000   20
60528   22
67000   19
71270   21
75016   27
78000   18
8022    23
80686   14
90110   32
Austria 55
B-6000  8
Belgium 25
COUNTRY 1
Canada  44
Denmark 63
EC2 5NT 51
FIN-02271       30
France  81
Italy   113
N 5804  29
Norway  32
Osaka   20
Philippines     26
S-958 22        19
Singapore       36
Sweden  38
T2F 8M4 26
UK      26
USA     969
WA1 1DP 12
WX1 6LT 29
WX3 6FW 26
neel@neel-Latitude-E6430:~/hadoop/sbin$
```

X. Comparison of output of Hive and MapReduce

A.Output using hive tables

According to product line:

```
+--------------------+---------------+----------+
|   product_line     | no_prod_line  | sum_qty  |
+--------------------+---------------+----------+
| Classic Cars       | 967           | 33992    |
| Motorcycles        | 331           | 11663    |
| PRODUCTLINE        | 1             | NULL     |
| Planes             | 306           | 10727    |
| Ships              | 234           | 8127     |
| Trains             | 77            | 2712     |
| Trucks and Buses   | 301           | 10777    |
| Vintage Cars       | 607           | 21069    |
+--------------------+---------------+----------+
8 rows selected (39.361 seconds)
0: jdbc:hive2://localhost:10000>
```

According to Country:

```
neel@neel-Latitude-E6430: ~/apache-hive-3.1.2-bin
neel@neel-Latitude-E64...  neel@neel-Latitude-E64...  neel@neel-Latitude-E64...
sec
INFO  : MapReduce Total cumulative CPU time: 4 seconds 220 msec
INFO  : Ended Job = job_1604676470957_0002
INFO  : MapReduce Jobs Launched:
INFO  : Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.22 sec   HDFS Read: 547441
HDFS Write: 1310 SUCCESS
INFO  : Total MapReduce CPU Time Spent: 4 seconds 220 msec
INFO  : Completed executing command(queryId=neel_20201106210658_110605ec-981f-4433-8f8f-
da20c33e5f61); Time taken: 27.441 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
+--------------+-------------------------+----------+
|   country    | no_of_sales_percountry  | sum_qty  |
+--------------+-------------------------+----------+
|              | 192                     | 6827     |
| 106-0032     | 32                      | 1150     |
| 1203         | 31                      | 1078     |
| 13008        | 25                      | 804      |
| 2            | 16                      | 490      |
| 2060         | 46                      | 1469     |
| 2067         | 46                      | 1601     |
| 21240        | 30                      | 1051     |
| 28023        | 13                      | 468      |
| 28034        | 259                     | 9327     |
| 3004         | 55                      | 1926     |
| 3150         | 23                      | 705      |
| 4101         | 15                      | 545      |
| 44000        | 60                      | 2102     |
| 50739        | 26                      | 936      |
| 59000        | 20                      | 699      |
| 60528        | 22                      | 811      |
| 67000        | 19                      | 779      |
| 71270        | 21                      | 668      |
| 75016        | 27                      | 1001     |
| 78000        | 18                      | 637      |
| 8022         | 23                      | 882      |
| 80686        | 14                      | 401      |
| 90110        | 32                      | 1110     |
| Austria      | 55                      | 1974     |
| B-6000       | 8                       | 278      |
| Belgium      | 25                      | 796      |
| COUNTRY      | 1                       | NULL     |
| Canada       | 44                      | 1420     |
| Denmark      | 63                      | 2197     |
| EC2 5NT      | 51                      | 1778     |
| FIN-02271    | 30                      | 1031     |
| France       | 81                      | 2804     |
| Italy        | 113                     | 3773     |
| N 5804       | 29                      | 973      |
| Norway       | 32                      | 1082     |
| Osaka        | 20                      | 692      |
| Philippines  | 26                      | 961      |
| S-958 22     | 19                      | 647      |
| Singapore    | 36                      | 1236     |
| Sweden       | 38                      | 1359     |
| T2F 8M4      | 26                      | 873      |
| UK           | 26                      | 895      |
| USA          | 969                     | 34491    |
| WA1 1DP      | 12                      | 357      |
| WX1 6LT      | 29                      | 1046     |
| WX3 6FW      | 26                      | 937      |
+--------------+-------------------------+----------+
47 rows selected (28.007 seconds)
0: jdbc:hive2://localhost:10000>
```

B.Output using MapReduce

According to product line:

```
neel@neel-Latitude-E6430:~/hadoop/sbin$ hdfs dfs -cat /sales_count_output/part-
00000
Classic Cars    967
Motorcycles     331
PRODUCTLINE     1
Planes  306
Ships   234
Trains  77
Trucks and Buses        301
Vintage Cars    607
```

According to Country:

```
neel@neel-Latitude-E6430:~/hadoop/sbin$ hdfs dfs -cat /sales_count_output_country/part-00000
        192
106-0032        32
1203    31
13008   25
2       16
2060    46
2067    46
21240   30
28023   13
28034   259
3004    55
3150    23
4101    15
44000   60
50739   26
59000   20
60528   22
67000   19
71270   21
75016   27
78000   18
8022    23
80686   14
90110   32
Austria 55
B-6000  8
Belgium 25
COUNTRY 1
Canada  44
Denmark 63
EC2 5NT 51
FIN-02271       30
France  81
Italy   113
N 5804  29
Norway  32
Osaka   20
Philippines     26
S-958 22        19
Singapore       36
Sweden  38
T2F 8M4 26
UK      26
USA     969
WA1 1DP 12
WX1 6LT 29
WX3 6FW 26
neel@neel-Latitude-E6430:~/hadoop/sbin$
```

C.Comparison

For product line:

The number of orders sorted according to product line is the same for both HIVE and MapReduce.

For country:

The number of orders sorted according to country is also shown same for both HIVE and MapReduce.

This shows that the HIVE framework is successfully able to perform mapreduce jobs and it is way more simpler to write HIVEQL than MapReduce jobs so HIVE is an effective framework made by Apache Software foundation.

XI. YAFFS2

A.Introduction

YAFFS2 is a log structured file system designed to function with NAND flash memory.YAFFS2 is extremely reliable for NAND flash memory devices.Program or erase failures are detected in the hardware itself.The charge leaks are handled by Error Correction Codes.Write disturb which means that sometimes extra bits are set to 0 are handled by YAFFS2 by not rewriting to pages [36].

It is compiler and Operating System neutral. It has a portable OS interface and an app interface.It is Log Structured in nature. Each activity is tracked by writing a log in the kernel logs. Size of each log is equal to the size of each chunk.The size of each chunk is configurable along with file limit and OOB layout.It is single threaded which means it has no separate thread for garbage collection like NOR flash.It strictly follows a no rewrite policy.YAFFS1 was deprecated since it was writing to the OOB area of the chunk to mark it as deleted[36].

A page in the NAND flash memory can be of 512B or 2048 B. A block is the unit of erasure in the NAND flash .And a page is a subunit of a block in the NAND flash. To correspond this methodology YAFFS2 has Blocks which correspond to Blocks of NAND flash and it has chunks that correspond to pages in NAND flash. This method was not available in any other file systems, therefore YAFFS2 was invented[36].

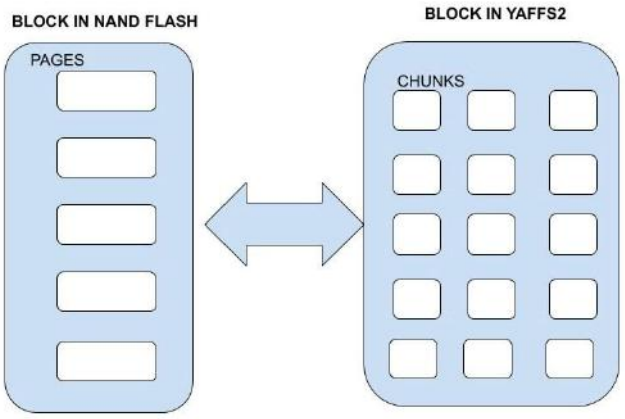


Figure 20. Similarities in NAND Flash and YAFFS2

B.Flash File Systems

Flash Memory is a non-volatile memory which means that even when the chip is not powered it has the data stored.The content stored in it is also modifiable[33]. It is an EEPROM type of memory . EEPROM stands for Electrically Erasable Read Only Memory.Mobile devices are quite portable so they have to be shockproof and lightweight.The traditional hard drives could not be compressed in size to fit in the mobile devices since it had rotors and cooling systems were required.A different approach had to be followed for mobile devices therefore an idea of using Flash memory came into play. Flash memory is shock resistant , it is a solid state storage which means there is no head pin rotating to access the data from the files from the tracks of the hard disk. It has a high density of storage for a low cost.The data is stored in blocks which are device dependent and have a fixed size. By default all the blocks are set to 1.When a write operation occurs the number changes from 1 to 0 or remains 1 . Different combinations of 0s and 1s makeup the whole data.To erase all data, all the blocks are set to 1 .But they can be rewritten only a limited number of times. The data is distributed evenly among all the blocks using the concept of wear leveling.[34]

NAND flash memory is used in embedded systems and the prices for nand flash devices are also low,but it has some limitations. Block erasure can only be done for a limited number of times. When writing or reading of data occurs there are chances of bit flips which results in errors in modification of data. The file systems that already exist for hard disk storage devices cannot be used for flash memory.[35].There are two types of Flash memories namely NAND flash and NOR flash.[36]

NOR flash memory is fully addressable which means each byte can be addressed individually. It is used to hold the system firmware, PC's BIOS(Basic Input Output System). It can also hold the entire software of an embedded system.It is slower than dynamic memory that is RAM.So on boot up the NOR flash memory metadata is copied into RAM for faster access. It suffers from huge write times and rewrites entire blocks at a time.

NAND flash is cheaper and denser.Data is not read or written randomly , content has to be read a page(chunk) at a time.Modification is also done a page at a time in erase-write cycles.Page size can be 512B or 2048B.It is a block device which is similar to a magnetic disk. A block is read at a time and there are limited cycles to read and write to NAND flash.

Characteristics	NOR flash	NAND flash
Acess mode	Linear random access	Page access
Replaces	ROM	Mass storage
Cost	Expensive	Cheap
Device Density	Low(64MB)	High(1GB)
Erase block size	8K-> 128K	32 x 512B/64 x 2K pages
Endurance	100k to 1M erasures	10k to 100k erasures
Erase time	1 sec	2ms
Programming	Byte by byte and no limitations on write	Page programming is used and the whole block is erased at a time .To write to a page it should be erased
Data Sense	Program the bytes to change 1s to 0s	Program Pages to change from 1 to 0
Erase	0s changes to 1s	0s changes to 1s
Write ordering	Random access programming	Pages must be written sequentially within a block
Bad blocks	None when manufactured but may occur due to wear . File system is responsible to handle it.	There may be bad blocks when delivered , file system must be highly fault tolerent to handle them.
OOB Data	No	Yes(16 bytes)

Table 1. Comparison between NAND flash and NOR flash

C.Requirements for YAFFS2

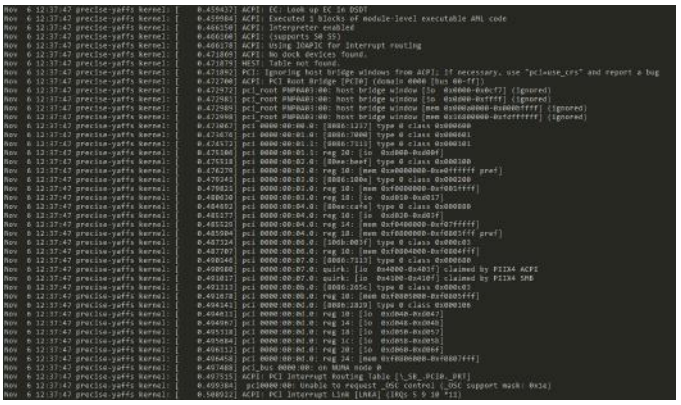
For implementing the YAFFS2 file system on linux , I have used the Vagrant Virtual Box provided by yaffs.net[36]. This virtual box comes with the kernel premounted. To emulate the NAND flash memory in this virtual box I have used the commands[36].

$ sudo sudo modprobe nandsim first_id_byte=0xec second_id_byte=0xd3 third_id_byte=0x51 fourth_id_byte=0x15

$ sudo mount -t yaffs /dev/mtdblock0 /media/nand

Log files:

cat /var/log/kern.log



D.Analysis of YAFFS2

Anything that happens in YAFFS2 is first an object.There are different types of objects and they are managed by YAFFS2.The different types of objects include data file, directory, hard link, soft link, special object like a pipe or a device file.Blocks are a unit of erasure and the data is stored in chunks.Blocks can be in different states like DEAD , SCANNING, UNKNOWN,FULL, EMPTY, ALLOCATING,DIRTY , COLLECTING ,CHECKPOINT.[36]. Prescanning phase includes UNKNOWN. Scanning phase involves NEEDS SCANNING AND PRE SCANNING. The regular runtime states are FULL, DIRTY, EMPTY, ALLOCATING.Garbage collection happens in COLLECTING state and checkpointing happens in CHECKPOINT state.

If a block is in FULL state then all the chunks are allocated.If a block is in EMPTY state then majority chunks are ready to be written. If a block is in ALLOCATING state then that block is selected for writing. If a block is in DIRTY state then the chunks are obsolete in nature for that block. If a block is in COLLECTING state then Garbage Collector is finding for currently written chunks and is copying them to another block to erase the current block. If a block is in CHECKPOINT state means that the block contains checkpoint data.

OOB area tag has header tag and object tag, it contains the block Sequence number blockState, Chunk ID, Object ID, nBytes,blockSequence, ECC(Error Correcting Codes).

0	7		9	9-15
tag 1	...tag 7	status 1	status 2	6 ECC tags

A chunk has the following structure .[34]

CHUNK

object ID	identifies object to which chunk it belongs to. It is uniques for every file header, file permissions, length
Chunk ID	shows where the chunk belongs in the object
Byte Count	Number of bytes of valid data in the file

Figure 21. CHUNK

This is how the logs are maintained in kernel logs[36]

Create a file					
Block	Chunk	Object ID	Chunk ID	Status	Content
0	0	499	0	Live	Oject header for file(length=0)
0	1				
0	2				
0	3				

After Write operations:					
Block	Chunk	Object ID	Chunk ID	Status	Content
0	0	499	0	Live	Oject header for file(length=0)
0	1	499	1	Live	First chunk
0	2	499	2	Live	Second chunk
0	3	499	3	Live	Third Chunk

After Process closes					
Block	Chunk	Object ID	Chunk ID	Status	Content
0	0	499	0	Deleted	Oject header for file(length=0)
0	1	499	1	Live	First chunk
0	2	499	2	Live	Second chunk
0	3	499	3	Live	Third Chunk
1	0	499	0	Live	Oject header for file(length=n)
1					
1					
1					

After reopening and modifications:					
Block	Chunk	Object ID	Chunk ID	Status	Content
0	0	499	0	Deleted	Oject header for file(length=0)
0	1	499	1	Deleted	First chunk
0	2	499	2	Live	Second chunk
0	3	499	3	Live	Third Chunk
1	0	499	0	Deleted	Oject header for file(length=n)
1	1	499	1	Live	New first chunk written
1	2	499	0	Live	Object header of file
1	3				

E.Garbage Collection and Wear leveling

The goal of flash memory should be to perform Wear leveling since there are a limited number of erase write cycles available for NAND flash the data should be distributed evenly among all the blocks.[34]

Wear leveling can be done implicitly or explicitly,but it is mainly done implicitly which means there is no special functionality assigned to do wear leveling, it happens automatically when other operations are taking place.There are two forms of wear leveling . One is dynamic wear leveling and the other is static wear leveling.

Dynamic Wear Leveling	Static Wear Leveling
There are high and low usage areas in memory. Both of them perform swap(high,low)	
controller maintains mapping of logical block addresses to the flash memory block addresses	it periodically moves the unmodified blocks i.e. static data to heavily used blocks
a block of data, instead of getting rewritten it gets written to a new block, blocks that do not get modified never get reused therefore the content remains the same	all blocks get a chance to be rewritten therefore they wear out evenly

Table 3. Dynamic Wear leveling vs Static wear leveling

Garbage collection has to be done because whenever a file gets modified,old object header data becomes obsolete and updated entries are written to the log .If all chunks in a block are garbage then the whole block is erased by resetting all the chunk bits to 1.There are two types of Garbage collection methods namely Aggressive garbage collection and Passive garbage collection. A block is analysed by the Garbage Collector(GC). Garbage collector always runs in the background.If it finds a block which has a majority chunks as invalid which means they are obsolete and then if it deletes the block then it is called Passive Garbage collection. But if the GC finds out that the majority of the chunks in a block are live and useful and still it erases the block then it is called Aggressive Garbage collection [33].GC has to be stopped for sometime if there is no high activity since then it will perform aggressive Garbage collection which is not good for the memory since it causes wear.

ACKNOWLEDGMENT

I would like to thank Mr.Vijayakumar K sir for approving our project topic Comparative study on file allocation methods, file systems in windows as well as Linux and mobile OS. Study on distributed file systems and multimedia file systems and for guiding us to study more research papers.

References

[1] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler Yahoo! Sunnyvale, California USA,"The Hadoop Distributed File System" by K Shvachko. https://storageconference.us/2010/Papers/MSST/Shvachko.pdf

[2] Shiva Asadianfam, Mahboubeh Shamsi and shahrad kashany ," A Review: Distributed File System ". http://www.ijcncs.org/published/volume3/issue5/p6_3-5.pdf

[3] L.Sudha Rani, K. Sudhakar , S.Vinay Kumar, Assistant Professor, Computer Science Department, GPREC, Kurnool, " A Survey of Distributed File Systems". https://www.cs.cmu.edu/~satya/docdir/satya89survey.pdf

[4] Darren Quick1, Mohammed Alzaabi, University of South Australia, Adelaide, Australia, " Forensic analysis of the android file system yaffs2". https://www.researchgate.net/publication/254592225_Forensic_analysis_of_the_android_file_system_YAFFS2

[5] Isma Irum, Mudassar Raza, Muhammad Sharif , " File Systems for Various Operating Systems: A Review". https://www.researchgate.net/publication/236898943_File_Systems_for_Various_Operating_Systems_A_Review

[6] Subham, "Android File System Explained"17thMarch '18 . (https://android.tutorials.how/android-file-system/)

[7] JEFF TYSON, "How flash memory works". (https://computer.howstuffworks.com/flash-memory.htm)

[8] Wikipedia, "exFAT".(https://en.wikipedia.org/wiki/ExFAT)

[9] Wikipedia,"F2FS".(https://en.wikipedia.org/wiki/F2FS)

[10] Wikipedia,"JFFS2"(https://en.wikipedia.org/wiki/JFFS2)

[11] Wikipedia,"List of File Systems"(https://en.wikipedia.org/wiki/List_of_file_systems#File_systems_optimized_for_flash_memory,_solid_state_media)

[12] Learnitguide.net,"Difference between ext2 ext3 and ext4 file system in Linux". August 07,2016.(https://www.learnitguide.net/2016/08/difference-between-ext2-ext3-and-ext4.html)

[13] Wikipedia,"Extended file systems".(https://en.wikipedia.org/wiki/Extended_file_system)

[14] Andrew S. Tanenbaum ,"Example File Systems".Mar 8,2002.(https://www.informit.com/articles/article.aspx?p=25878&seqNum=4)

[15] Christensson, P. (2006). VFAT Definition. Retrieved 2020, Aug 29, from https://techterms.com/definition/vfat

[16] linux.die.net"proc(5)-Linux man page".(https://linux.die.net/man/5/proc)

[17] kernel.org "Cgroups".(https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt)

[18] tldp.org ,"Linux System Administrator's Guide".(https://tldp.org/LDP/sag/html/root-fs.html)

[19] kernel.org,"Tmpfs".(https://www.kernel.org/doc/html/latest/filesystems/tmpfs.html)

[20] Shashank Jain,"Pseudo File Systems in Linux".Aug 11,2018 .(https://medium.com/@jain.sm/pseudo-file-systems-in-linux-5bf67eb6e450)

[21] man7.org ,"sysfs(5)- Linux manual page" . (https://man7.org/linux/man-pages/man5/sysfs.5.html)

[22] Charles Manning,"How YAFFS Works".2007-2010.(http://dubeyko.com/development/FileSystems/YAFFS/HowYaffsWorks.pdf)

[23] aalto university ace ,"Q & A session with Linus Torvalds: Why is Linux not competitive on desktop?".15 Jun 2012.(https://www.youtube.com/watch?v=KFKxlYNfT_o)

[24] hadoop.apache.org ,"HDFS Architecture Guide".(https://hadoop.apache.org/docs/r1.2.1/hdfs_design.pdf)

[25] netmeister.org,"Chapter 4 Of File Systems and Storage Models".pp.106 to 110.(https://www.netmeister.org/book/04-file-systems.pdf)

[26] B Depardon - 2013 ,"Analysis of Six Distributed File Systems".Feb 15 2013.(https://hal.inria.fr/hal-00789086/file/a_survey_of_dfs.pdf)

[27] Wikipedia,"Journaling file system".(https://en.wikipedia.org/wiki/Journaling_file_system)

[28] Wikipedia,"Network File System".(https://en.wikipedia.org/wiki/Network_File_System)

[29] Tutorials point. (n.d.). Hive Tutorial. Hive Tutorial - Tutorials Point. https://www.tutorialspoint.com/hive/index.htm

[30] White, T. (2010). Hadoop: The definitive guide. Sebastopol: O'Reilly Media. https://www.oreilly.com/library/view/hadoop-the-definitive/9780596521974/

[31] Datareportal, Simon Kemp, 30-01-2020.https://datareportal.com/reports/digital-2020-global-digital-overview

[32] Udacity, Dough Cutting: The Origins of Hadoop,Feb 24 2015. https://www.youtube.com/watch?v=ebgXN7VaIZA

[33] pk.org File System Design Case Studies: File Systems for Flash Memory.https://www.cs.rutgers.edu/~pxk/416/notes/13a-fs-studies.html

[34] Zimmermann, Christian. (2011). Mobile Phone Forensics: Analysis of the Android Filesystem (YAFFS2). https://www.researchgate.net/publication/334645476_Mobile_Phone_Forensics_Analysis_of_the_Android_Filesystem_YAFFS2

[35] elinux.org Evaluation of Flash File Systems for Large NAND Flash Memory. https://elinux.org/images/7/7e/ELC2009-FlashFS-Toshiba.pdf

[36] Aleph One Ltd. YAFFS A NAND flash filesystem. https://yaffs.net/

[37] kaggle.com Gus Segura .Sample Sales Data. https://www.kaggle.com/kyanyoga/sample-sales-data