Unit 8: File System

8.2. The Windows 2000 File System (NTFS)

Windows 2000 File System - Terminology

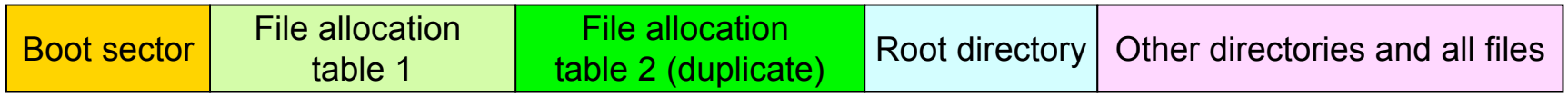
- Sectors:
  - hardware-addressable blocks on a storage medium
  - Typical sector size on hard disks for x86-based systems is 512 bytes
- File system formats:
  - Define the way data is stored on storage media
  - Impact a file system features: permissions & security, limitations on file size, support for small/large files/disks
- Clusters:
  - Addressable blocks that many file system formats use
  - Cluster size is always a multiple of the sector size
  - Cluster size tradeoff: space efficiency vs. access speed
- Metadata:
  - Data stored on a volume in support of file system format management
  - Typically not accessible to applications

Windows 2000 File System Formats – CDFS, UDF

- CDFS:
  - CD-ROM file system defined in 1988 (legacy, superseded by UDF)
  - ISO 9660 Level 2 support for long filenames
  - Implemented in \Winnt\System32\Drivers\Cdfs.sys
  - Directory and file names must be fewer than 32 characters long
  - Directory trees can be no more than eight levels deep
- UDF:
  - ISO 13346-compliant implementation of Universal Disk Format (UDF)
  - UDF 1.02 and 1.5 supported as defined by Optical Storage Technology Association (OSTA) in 1995
  - Filenames can be 255 characters long, case-sensitive
  - Maximum path length is 1023 characters
  - \Winnt\System32\Drivers\Udfs.sys provides read-only support

Windows 2000 File System Formats – FAT12, FAT16, FAT32

- FAT:
  - Primarily supported for compatibility with other OS in multiboot systems, and as a floppy disk format
  - Implemented in \Winnt\System32\Drivers\Fastfat.sys
  - Write-through semantics
  - 12-bit, 16-bit, 28-bit (FAT32) cluster identifier
  - FAT16: cluster size varies between 512 byte (< 32 MB volume size) and 64 Kbytes (> 2 GB volume size)
  - Windows 2000 uses FAT12 if volume size is less than 16 MB
  - Root directory of FAT12/16 volumes are pre-assigned enough space to store 256 entries (at a pre-defined cluster location on disk)
  - Maximum file size is 4GB



FAT format organization

Windows NT File System (NTFS)

Design Goals and Features:

- Overcome limitations inherent in FAT / HPFS
  - FAT (File Allocation Table) does not support large disks very well
  - FAT16 (MS-DOS file system) supports only up to 2^{16} clusters and 2 GB disks (with 64 Kb clusters!!)
  - FAT / root directory represents single point of failure
  - Number of entries in root directory is limited

  - HPFS removed some of FAT‘s limitations, but still did not support recoverability, security, data redundancy, and fault-tolerance (later versions of HPFS support up to 2TeraByte disks)

NTFS Recoverability

PC disk I/O in the old days: Speed was most important

NTFS changes this view – Reliability counts most:

- I/O operations that alter NTFS structure are implemented as atomic transactions
  - Change directory structure,
  - extend files, allocate space for new files
- Transactions are either completed or rolled back
- NTFS uses redundant storage for vital FS information
  - Contrasts with FAT / HPFS on-disk structures, which have single sectors containing critical file system data
  - Read error in these sectors -> volume lost

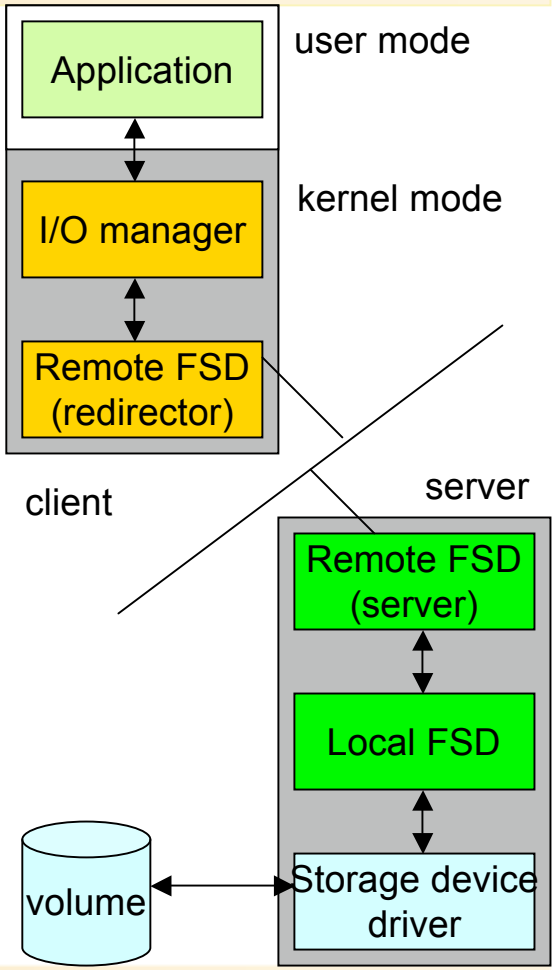
File System Driver Architecture

- Local File System Drivers (Local FSDs):
  - Ntfs.sys, Fastfat.sys, Udfs,sys, Cdfs,sys
  - Responsible for registering with the I/O manager and volume recognition/integrity checks
  - FSD creates device objects for each mounted file system format
  - I/O manager makes connection between volume‘s device objects (Created by storage device) and the FSD‘s device object
  - Local FSDs use cache manager to improve file access performance
  - Dismount operation permits the system to disconnect FSD from volume object
    - When media is changed or when application requires raw device access
    - I/O manager reinitiated volume mount operation on next access to media

File System Driver Architecture (contd.)

Remote File System Drivers (Remote FSDs):

- Client-side FSD translates I/O requests from applications into network file system protocol commands
- Server-side FSD listens for network commands and issues I/O requests to local FSD
- Windows 2000 client-side remote FSD: LANMan Redirector
  - Implemented as port/miniport driver
  - Includes Win32 service Workstation
- Server-side FSD server: LANMan Server
  - Includes Win32 service Server
  - CIFS – common internet file system (enhancement of Server Message Block protocol)



NTFS Security and Recoverability

NTFS security is derived from Windows 2000 object model

- Open file is implemented as file object;
  security descriptor is stored on disk as part of the file
- NT security system verifies access rights when a process tries to open a handle to any object
- Administrator or file owner may set permissions

NTFS recoverability ensures integrity of FS structure

- No guarantees for complete recovery of user files
- Layered driver model + FTDISK driver
  - Mirroring of data – RAID level 1
  - Striping of data - RAID level 5 (one disk with parity info)

Large Disks and Large Files

- Efficient support for large files and disks in NTFS
- FAT16:
  - 16-bit wide table stores allocation status of disk
  - Up to 65.536 clusters per volume (#files !!); adjustable cluster size
- FAT32:
  - New in Windows 2000
  - 4kb clusters on volumes up to 8 GB
  - Can relocate root directory / use backup copy of FAT
  - Root directory is ordinary cluster chain – no limits on #entries
- HPFS (support dropped in NT 4.0):
  - 32 bits to enumerate allocation units; maximum file size: 4GB
  - Allocates disk space in terms of physical sectors of 512 bytes; problem with some disks (1024 bit sectors)

Large Disks and Large Files (contd.)

- NTFS enumerates cluster with 64-bit numbers
- Up to 2^{64} clusters of up to 64 Kbytes size
- Maximum file size: 2^{64} bytes
- Cluster size is adjustable
  - 512 bytes on small disks
  - Maximum of 64Kb on large disks
- Multiple data streams
  - File info: name, owner, time stamps, type implemented as attribute
  - Each attribute consists of a stream – sequence of bytes
  - Default data stream has no name
  - New streams can be added: myfile.dat:stream2
  - File operations manipulate all streams simultaneously

Used to implement services for Macintosh in Windows NT Server

Additional Features in NTFS

- Unicode–based names
  - Each directory and filename may have 255 Unicode characters
- General indexing facility
  - NTFS supports indexing of file attributes on a disk volume (efficient sort-by-attribute, matching)
  - NTFS in Windows 2000 can index object IDs for files (enterprise-wide unique identifiers for files)
- Dynamic bad-cluster re-mapping
  - Read access to bad cluster fails; NTFS replaces the cluster/no re-use
  - FTDISK.SYS driver may retrieve good copy of data
- POSIX support
  - Hard links, „file-change-time“ time stamp
  - Case-sensitive file and directory names
  - Use Win32 CreateHardLink() or POSIX ln() function

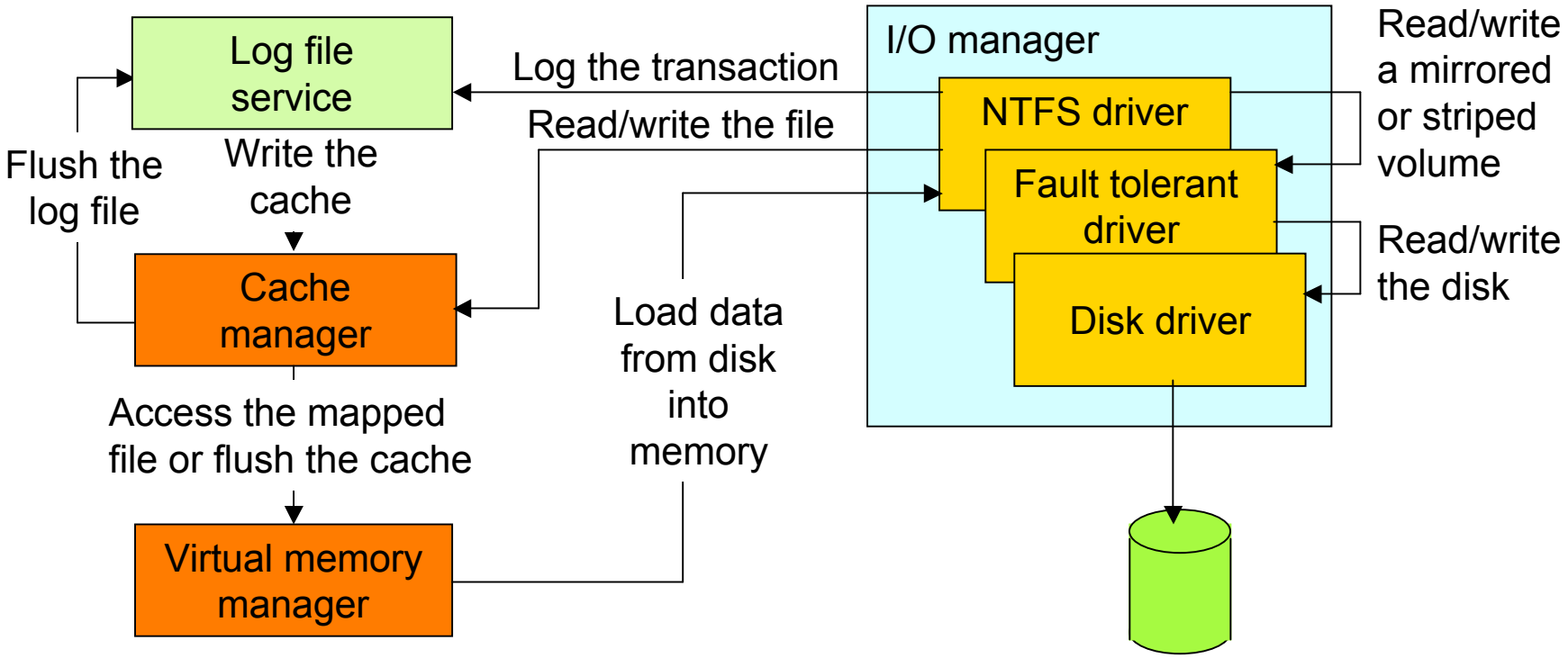
Junctions (symbolic links)

- Junctions allow a directory to redirect file/directory pathname translations to an alternate directory
  - Junctions can only link to local volumes
- Junctions are based upon reparse points
  - File or directory with user-defined reparse data
  - File system filter driver or I/O manager may evaluate reparse data
  - Each reparse point has a unique reparse tag
- Reparse tag owner can:
  - Manipulate the pathname specified in an I/O operation and continue I/O operation with altered pathname
  - Reparse tag owner may remove reparse tag, alter the file and re-issue the I/O operation (Used by Hierarchical Storage Management – HSM – to move files to tape and leave reparse point on disk)

Additional NFTS Features (contd.)

- Compression and Sparse Files
- Change Logging
  - Application can configure NTFS journal facility via DeviceIoControl(FSCTL_CREATE_USN_JOURNAL)
  - NTFS records file/dir changes to internal *change journal* file
  - obtain records via DeviceIoControl(FSCTL_QUERY_USN_JOURNAL)
- Per-User Volume Quotas
  - based on logical file sizes (compression doesn‘t help)
  - IDiskQuotaCOntrol, IDiskQuotaUser, IDiskQuotaEvents COM interfaces
- Link Tracking
  - Symbolic links that update automatically (based on Object IDs)
- Encryption

NTFS File System Driver



Components related to NTFS

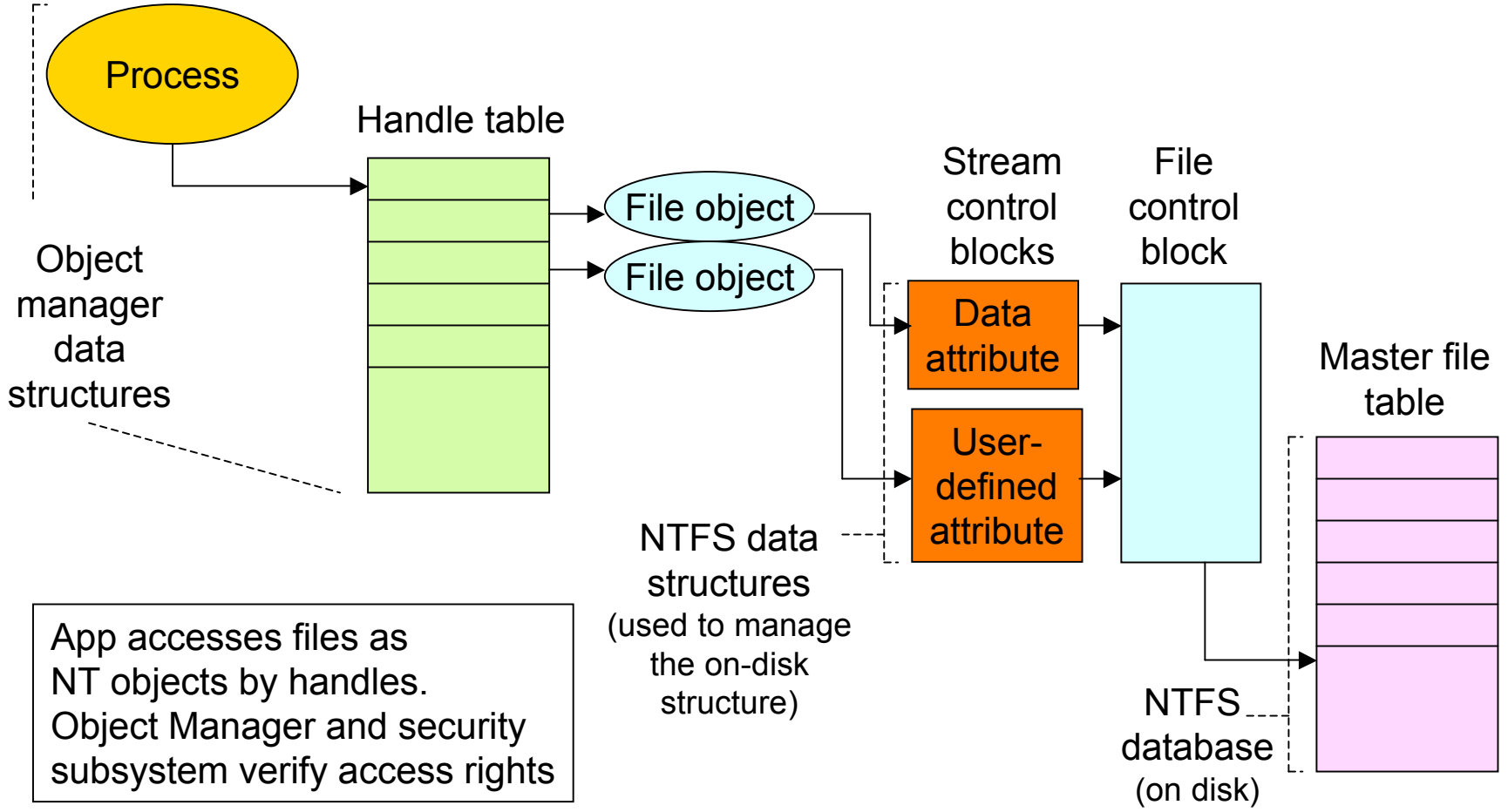
Cache Manager

- System wide caching
  - for NTFS and other file systems drivers
  - Including network file system drivers (server and redirectors)
- Cached files are mapped into virtual memory
  - Specialized interface from Cache Manager to NT virtual memory manager
  - Memory manager calls NTFS to access disk driver and obtain file

Log File Service

- 2 copies of transaction logs
- Transaction log is flushed to disk before write-data is sent to disk
- Cache manager performs actual flush operation

NTFS & File Objects



App accesses files as NT objects by handles. Object Manager and security subsystem verify access rights

NTFS On-Disk Structure

- Volumes correspond to logical partitions on disk
- Fault tolerant volumes may span multiple disks
  - Windows 2000 Disk Administrator utility
- Volume consists of series of files + unallocated space
  - FAT volume: some areas specially formatted for file system
  - NTFS volume: all data are stored as ordinary files
- NTFS refers internally to clusters
  - Cluster factor: #sectors/cluster; varies with volume size; (integral number of physical sectors; always a power of 2)
- Logical Cluster Numbers (LCNs):
  - refer to physical location
  - LCNs are contiguous enumeration of all clusters on a volume

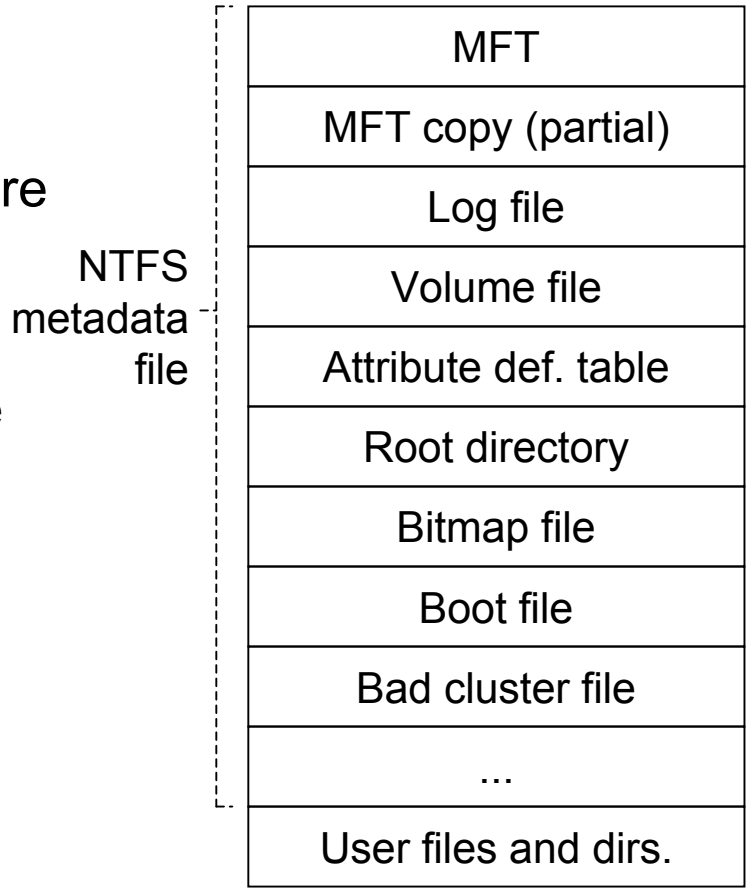
NTFS Cluster Size

- Default cluster size is disk-size dependent
  - 512 bytes for small disks (up to 512 MB)
  - 1 KB for disks up to 1 GB
  - 2 KB for disks between 1 and 2 GB
  - 4 KB for disks larger than 2 GB
- Tradeoff: disk fragmentation versus wasted space
- NTFS refers to physical locations via LCNs
  - Physical cluster = LCN * cluster-factor
- Virtual Cluster Numbers (VCNs):
  - Enumerates clusters belonging to a file; mapped to LCNs
  - LCNs are not necessarily physically contiguous

Master File Table

All data stored on a volume is contained in a file

- MFT: Heart of NTFS volume structure
  - Implemented as array of file records
  - One row for each file on the volume (including one row for MFT itself)
  - Metadata files store file system structure information (hidden files; $MFT; $Volume...)
  - More than one MFT record for highly fragmented files
  - Nfi.exe Utility from OEM Support Tools allows to dump MFT content (see support.microsoft.com/support/kb/articles/Q253/0/66.asp)



NTFS operation

Mounting a volume

1. NTFS looks in boot file for physical address of MFT ($MFT)
2. 2nd entry in MFT points to copy of MFT ($MFTMirr)
   - used to locate metadata files if MFT is corrupted
3. MFT entry in MFT contains VCN-to-LCN mapping info
4. NTFS obtains from MFT addresses of metadata files
   - NTFS opens these files
5. NTFS performs recovery operations
6. File system is now ready for user access

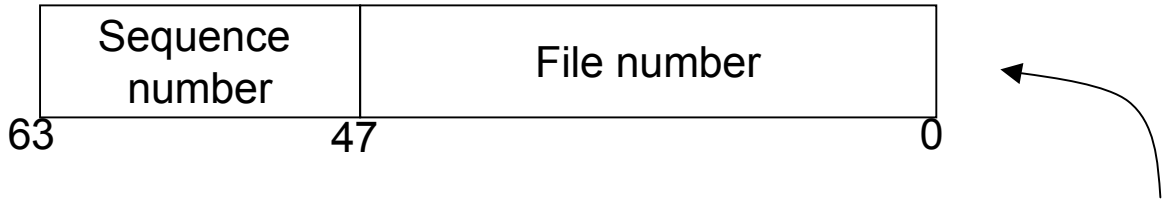
NTFS metadata

- NTFS writes to log file ($LogFile)
  - Record all commands that change volume structure
- Root directory:
  - When NTFS tries to open a file, it starts search in the root directory
  - Once the file is found, NTFS stores the file‘s MFT file reference
  - Subsequent read/write ops. may access file‘s MFT record directly
- Bitmap file ($Bitmap):
  - stores allocation state volume; each bit represents one cluster
- Boot file ($Boot):
  - Stores bootstrap code
  - Has to be located at special disk address
  - Represented as file by NTFS -> file ops. possible (!) (no editing)

NTFS metadata (contd.)

- Bad-cluster file ($BadClus)
  - Records bad spots on the disk
- Volume file ($Volume)
  - Contains: volume name, NTFS version
  - Bit, which indicates whether volume is corrupted
- Attribute Definition Table ($AttrDef)
  - Defines attribute types supported on the volume
  - Indicates whether they can be indexed, recovered, etc.

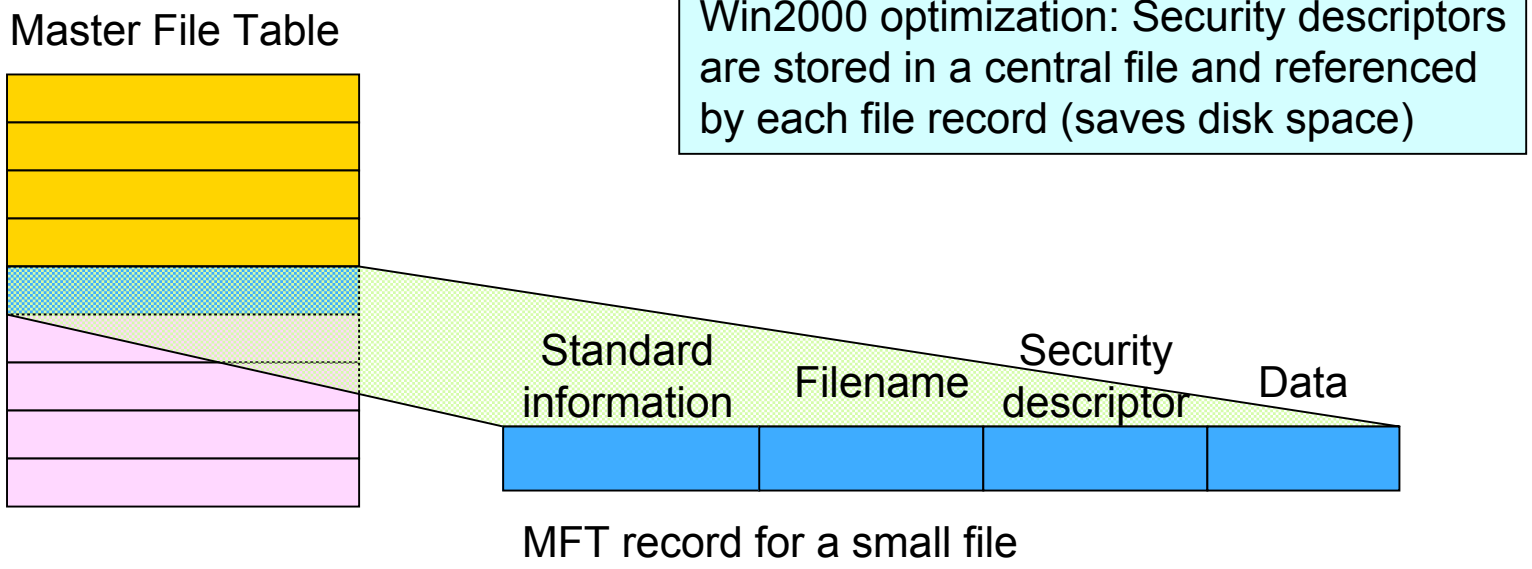
File Records & File Reference Numbers



- File on NTFS volume is identified by **file reference**
  - File number == index in MFT
  - Sequence number – used by NTFS for consistency checking; incremented each time a reference is re-used
- File Records:
  - File is collection of attribute/value pairs (one of which is data)
  - Unnamed data attribute
  - Other attributes: filename, time stamp, security descriptor,...
  - Each file attribute is stored as separate stream of bytes within a file

File Records (contd.)

- NTFS doesn‘t read/write files:
  - It reads/writes attribute streams
  - Operations: create, delete, read (byte range), write (byte range)
  - Read/write normally operate on unnamed data attribute



Standard Attributes for NTFS Files

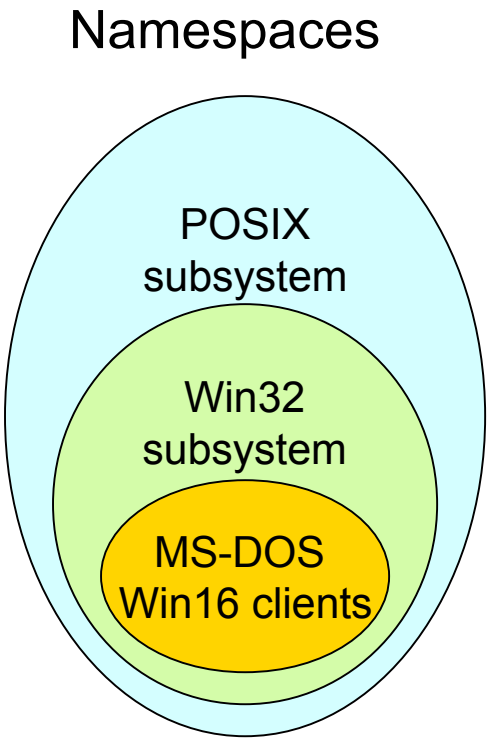
Attribute	Description
Standard information	File attributes: read-only, archive, etc; time stamps; creation/modification time; hard link count
Filename	Name in Unicode characters; multiple filename attributes possible (POSIX links!!); short names for access by MS-DOS and 16-bin Win applications
Security descriptor	Specifies who owns the file and who can access it
data	Contents of the file; a file has one default unnamed data attribute; directory has no default data attrib.
Index root, index	Three attributes used to implement filename allocation, bitmap index for large directories (dirs. only)
Attribute list	List of attributes that make up the file and first reference of the MFT record in which the attribute is located (for files which require multiple MFT file records)

Attributes (contd.)

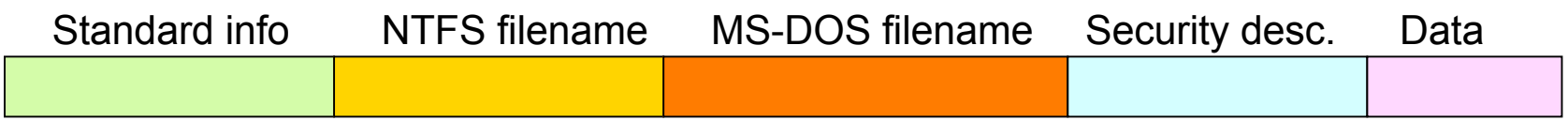
- Each attribute in a file record has a name and a value
- NTFS identifies attributes:
  - Uppercase name starting with $: $FILENAME, $DATA
- Attribute‘s value: Byte stream
  - The filename for $FILENAME
  - The data bytes for $DATA
- Attribute names correspond to numeric typecodes
- File attributes in an MFT record are ordered by typecodes
  - Some attribute types may appear more than once (e.g. Filename)

Filenames

- POSIX:
  - Case-sensitive, trailing periods & spaces
  - NTFS namespace equiv. to POSIX space
- Win32:
  - Long filenames, unicode names
  - Multiple dots, embedded spaces, beginning dots
- MS-DOS:
  - 8.3 names, case does not matter
- NTFS generates MS-DOS names for Win32 files automatically
  - Fully functional aliases for NTFS names
  - Stored in same directory as long names; dir /x



MS-DOS filenames in NTFS

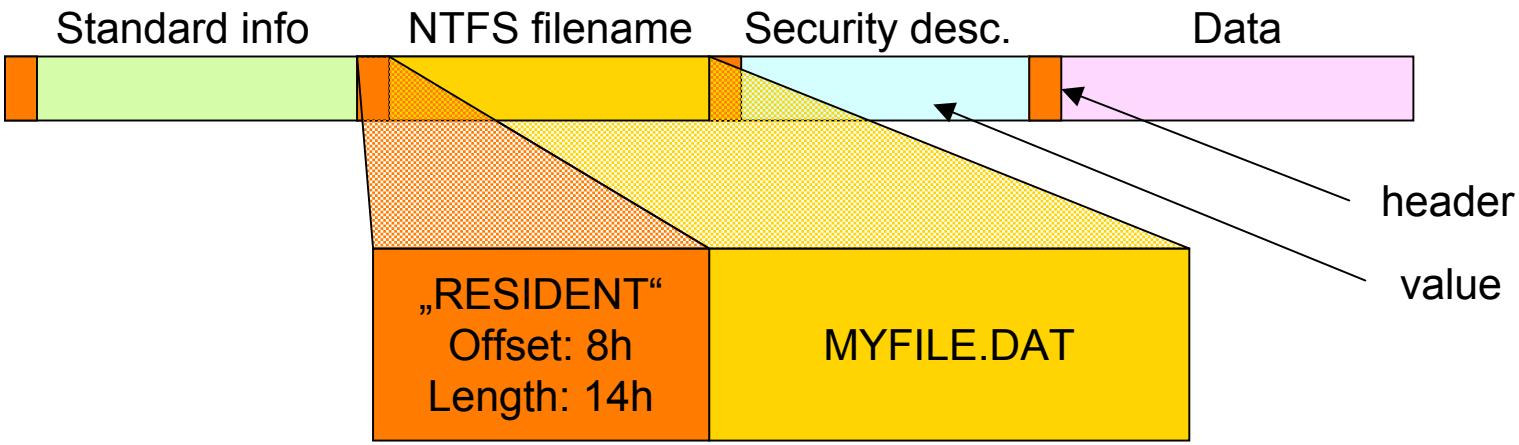


MFT file record with MS-DOS filename attribute

- NTFS name and MS-DOS name are stored in same file record and refer to same file
  - Renaming changes both filenames
  - Open, read, write, delete work with both names equally
- POSIX hardlinks are implemented in similar way
  - Deleting a file with multiple names only decreases link count
- Generation of MS-DOS names:
  1. Remove all illegal chars; remove all but one period; truncate to 6 chars
  2. Append ~1 to name; truncate extension to 3 chars; all uppercase
  3. Increment ~1 if filename duplicates an existing name in directory

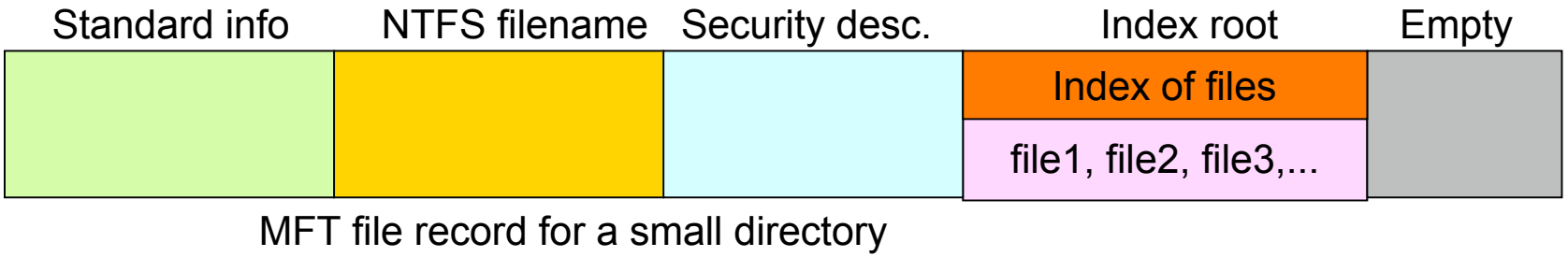
Resident & Nonresident Attributes

- Small files:
  - All attributes and values fit into MFT
  - Attribute with value in MFT is called „resident“
  - All attributes start with header (always resident)
  - Header contains offset to attr. value and length of value



Attributes (contd.)

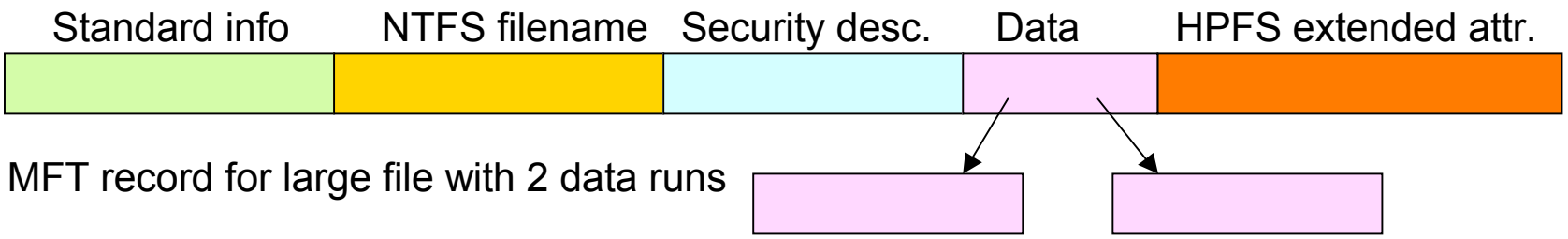
- Small directory:
  - index root attribute contains index of file references for files and subdirectories



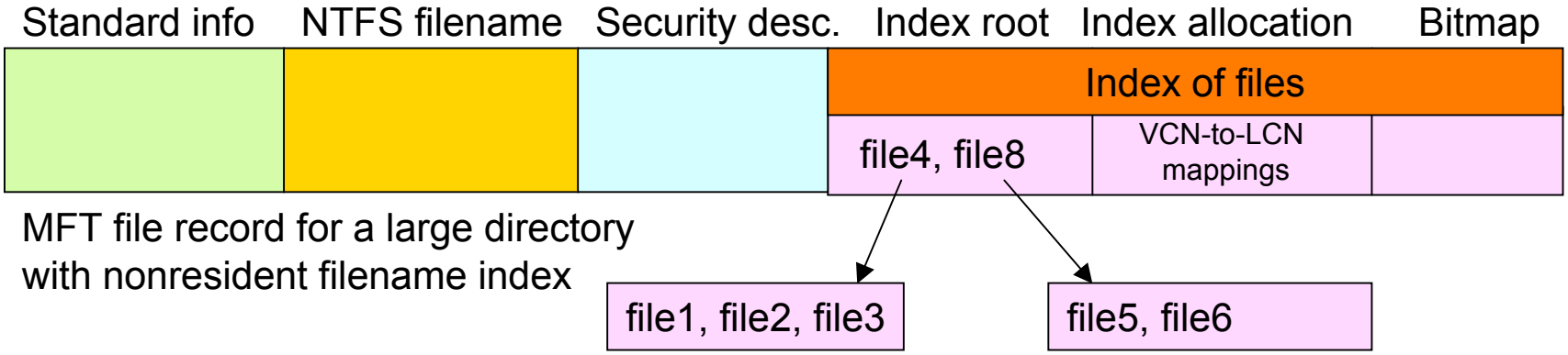
MFT file record for a small directory

- If file attribute does not fit into MFT:
  - NTFS allocates separate cluster (run, extent) to store the values
  - NTFS allocates additional runs if an attribute‘s value later grows
  - Those attributes are called „non-resident“
  - Header of non-resident attribute contains location info

Large files & directories

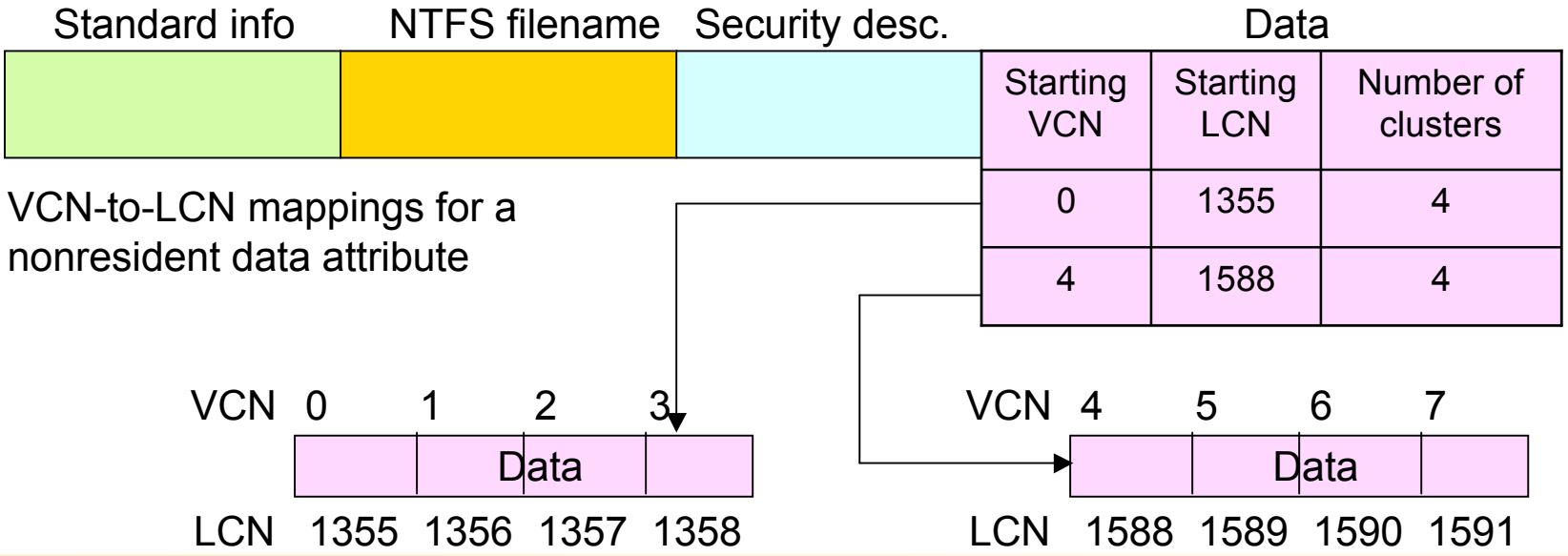


- Only attributes that can grow can be non-resident
- Filename & standard info are always resident
- Index of files for directories forms B+ tree



Large files (contd.)

- NTFS keeps track of runs by means of VCN (Virtual Cluster Numbers)
  - Logical Cluster Numbers represent an entire volume
  - Virtual Cluster Numbers represent clusters belonging to one file
  - Attribute lists may extend over multiple runs (not only data)



Data Compression

- NTFS supports compression
  - Per-file, per-directory, per-volume basis
  - NTFS compression is performed on user data only, not NTFS metadata

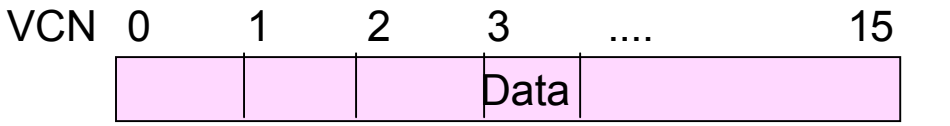
- Inspect files/volume via Win32:
  GetVolumeInformation(), GetCompressedFileSize()
- Change settings for files/directories:
  DeviceIoControl()

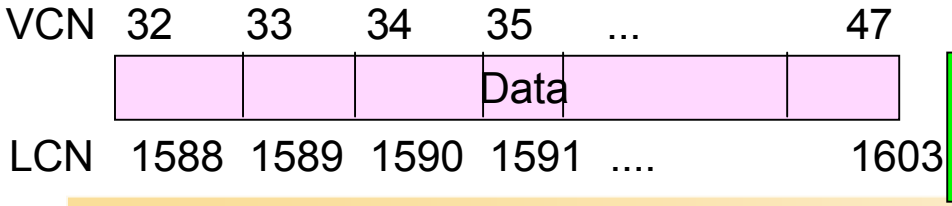
  with flags
  FSCTL_GET_COMPRESSION, FSCTL_SET_COMPRESSION

Compression of sparse files

- NTFS zeroes all file contents on creation (C2 req.)
- Many sparse files contain large amount of zero-bytes
  - These bytes occupy space on disk – unless files are compressed

Starting VCN	Starting LCN	Number of clusters
0	1355	16
32	1588	16
48	96	16
128	324	16



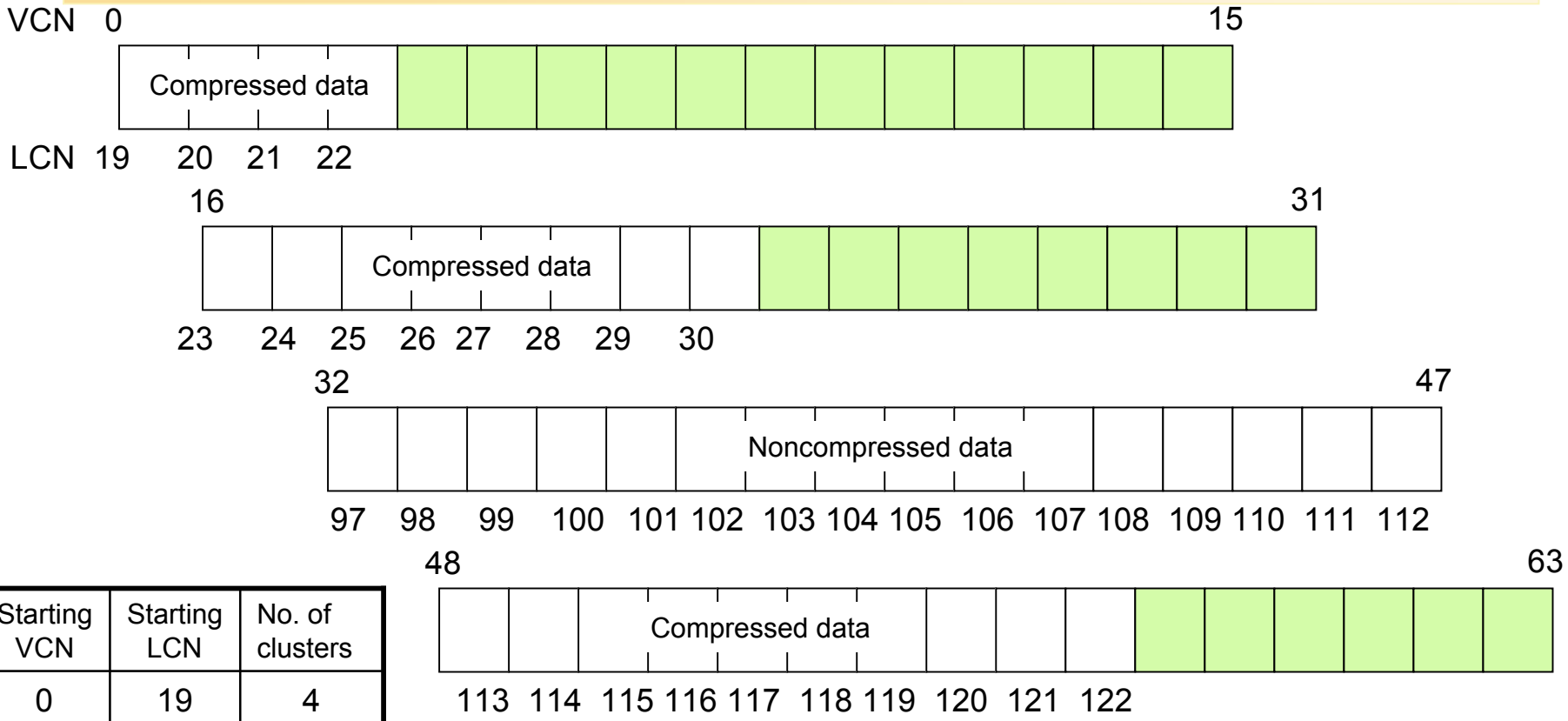


Certain ranges of VCNs have no disk allocation (16-31, 64-127)

Compressing Nonsparse Data

- NTFS divides the file‘s unprocessed data into *compression units* 16 clusters long
- Certain sequence might not compress much
  - NTFS determines for each compression unit whether it will shrink by at least on cluster
  - If data does not compress, NTFS allocates cluster space and simply writes data
  - If data compresses at least one cluster, NTFS allocates only the clusters needed for compressed data
- When writing data, NTFS ensures that each run begins on virtual 16-cluster boundary
  - NTFS reads/writes at least one compression unit when accessing a file
  - Read-ahead + asynch. decompression improves performance

Data runs of a compressed file



Starting VCN	Starting LCN	No. of clusters
0	19	4
16	23	8
32	97	16
48	113	10

MFT record for a compressed file

Windows 2000
NTFS Extensions

- Disk quotas on per-user bases
- Security descriptors (ACLs) can be stored once but referenced in multiple files
- Native support for properties (OLE), including indexing
- Reparse points – implementation of symbolic links
  - Mount points for arbitrary file system volumes
- Support for sparse files
- Distributed link tracking (via global object Ids)
  - Renaming the target file will no longer break links (shortcuts...)
- Add disk space to an NTFS volume without reboot
- No decompressing when transmitting files over network