Operating Systems (UE18CS302)
NTFS

Aronya Baksy

December 14, 2021

1 Introduction

- NTFS is a replacement for the older and widely used FAT-32 file system that is used by a wide variety of systems and other peripherals.
- Disadvantage of FAT-32 is that it does not restrict file system access to authorized users. This can be solved by encrypting files before storing in file system, but this leads to increased encryption overheads.
- NTFS uses Access Control Lists (ACLs) to control access to individual files and supports implicit encryption of individual files or entire volumes.
- NTFS implements many other features as well, including data recovery, fault tolerance, very large files and file systems, multiple data streams, UNICODE names, sparse files, journaling, volume shadow copies, and file compression.

2 Internal Layout

- Fundamental entity of NTFS is a volume (created by Windows logical disk mgmt utility, based on a logical disk partition). Volume may occupy a part of a disk, an entire disk, or span multiple disks.
- Disk space is allocated in **cluster** units (cluster is a number of sectors on disk). Larger cluster size indicates better performance for larger files in use nowadays, but can lead to internal fragmentation.
- Each cluster has a **logical cluster number** (LCN) (from 0 to N). Physical offset on disk can be calculated by doing $LCN \times cluster\_size$.
- Files in NTFS are structured objects with (attr, value) pairs. Each attribute is a byte stream than can be created/read/updated/deleted.
- Standard attributes for all files are name (and aliases if any), creation time, and descriptor that attaches the ACL.
- The data of the file is stored in an unnamed data attribute. Additional data streams can be created (eg: Macintosh files stored on Windows Server, The IProp interfaces of the Component Object Model (COM)). The size of the unnamed attribute is returned as file size when queried.
- Each file in NTFS has an entry in an array called the **Master File Table** (MFT). Small attributes of file are stored on the MFT record, and larger ones are stored on contiguous extents of the disk.
- The MFT record stores pointers to extents on disk. If one MFT record is not large enough for all the attributes, or the file is highly fragmented and needs many pointers, then one **base file record** is allocated which stores pointers to the overflow MFT records.
- Each file has an unique 64-bit ID called the **file reference** (48 bit file number, 16 bit sequence number). The 48-bit file number is the index of that file's record in the MFT.
- The sequence number is incremented every time an MFT entry is reused. The sequence number enables NTFS to perform internal consistency checks, such as catching a stale reference to a deleted file after the MFT entry has been reused for a new file.

2.1 Namespace Organization

- Hierarchy of directories, each directory is stored as a **B+ Tree**.
- The index root of a directory contains the root of the B+ tree, or pointer to disk extent that contains the rest of the B+ Tree.
- Directory entries contain name of file as well as attrs like file size and update time taken from the MFT entry of that file. This is done so that listings can be efficiently generated.

2.2 Metadata

- Metadata of NTFS is stored in files (first file is MFT, second file is first 16 entries of MFT copied, for fault tolerance reasons).
- Other metadata files are:
  1. **Log File** records all metadata updates
  2. **Volume File** contains volume name, version of NTFS, and a corrupt or not flag.
  3. **Attribute-definition table** indicates which attribute types are used in the volume and what operations can be performed on each of them.
  4. **Root** is the top level directory in the hierarchy.
  5. **Bitmap** is a binary array indicating free/allocated clusters.
  6. The **boot file** contains the startup code for Windows and must be located at a particular disk address so that it can be found easily by the ROM bootstrap loader. The boot file also contains the physical address of the MFT.
  7. **Bad cluster** file tracks bad areas on volume, used for error recovery.

3 Recovery

- File systems like UFS store metadata on disk, and they recover from crashes by using the fsck program to check all the file-system data structures and restore them forcibly to a consistent state.
- Restoring them often involves deleting damaged files and freeing data clusters that had been written with user data but not properly recorded in the metadata structure. This checking is slow and may cause large amount of data loss.
- In NTFS all file system data-structure operations are done using transactions.
- Before starting the operation, the transaction writes a log record containing undo and redo information. After the update is done the transaction writes a commit status message indicating success.
- Recovery is done by processing this log file, redoing all operations that were successfully completed and then undoing all operations that did not successfully finish.
- Every 5 seconds, a checkpoint record is written to the log. This removes the need for all logs before that time, and they can be deleted (to ensure that logs don't become too big).
- This ensures that metadata is in a consistent state but offers no guarantee on contents of user's data.
- The log is stored in the third metadata file at the beginning of the volume, with two parts: the logging area, which is a circular queue of log records, and the restart area (duplicated twice), which holds context information, such as the position in the logging area where NTFS should start reading during a recovery.
- The logging functionality is provided by the log-file service. This service takes care of logging and recovery, and also free space management of log file
- If free space in log file becomes low, the manager queues all incoming I/O requests, finishes servicing the existing ones and cleans the log file before resuming operation of new requests.

4 Security

- Security is enforced similarly to Windows object model.
- Each file references a security descriptor that contains the ACL for that file. For memory efficiency, these can be made into indirect references to shared objects for shared permissions between many files.
- Directory searching is done either one level at a time (to allow for access control and POSIX compliance but slower) or at one shot using longest prefix matching (fast but does not take permissions into account).

5 Volume Management

- Combination of disk partitions can be done using **volume sets**. A volume with NTFS can be combined easily, by simply extending the bitmap over to the new volume.
- NTFS continues to use the same LCN mechanism that it uses for a single physical disk, and the FtDisk driver supplies the mapping from a logical-volume offset to the offset on one particular disk.
- Physical partitions can also be combined using **stripe sets**. Assuming the stripe size to be x, then the first x bytes of one file are stored on one partition, the next x bytes in the next partition and so on. The allocation wraps around once all the partitions have been exhausted.
- This technique creates one large logical volume but increases I/O bandwidth due to the implied parallelism in this arrangement. NTFS also supports RAID 1 (mirroring) and RAID 5 (block-level striping with distributed parity).