CSE451 NTFS Variations and other File System Issues Spring 2001

Gary Kimura

Lecture #24

May 18, 2001

Today's Topics

- NTFS Variations
- Consistency and persistence
- Utilities
- Implementation Issues
- Memory mapped files and software caching

A Brief history of NTFS

- In its early years (before official public release) Windows NT only supported FAT/DOS and HPFS (from OS/2)
- NTFS was designed with features tailored specifically for NT
- It was done originally by 4 software engineers who had earlier implemented FAT and HPFS for NT. These four developers also did the cache manager and major parts of the NT's kernel mode runtime libraries
- NTFS has sprouted additional features since its release in 1993. Most of these new features have been with forward compatible

NTFS Variations

- The basic model is that everything is a file
- The master file table (MFT) describes each file on the volume including itself
- Bitmap for allocation
- Retrieval pointer information is stored in a compact form
- Directories are B+ trees
- Recoverable meta-data using a logging file
- Hard Links and reparse Points
- Compressed and spare data files

Where to start on an NTFS disk

- The NTFS volume starts with a boot sector at LBN=0, and a duplicate boot sector at LBN=(number of sectors on the partition div 2).† So a disk with N sectors start with two boot sectors as illustrated.

```
  0             ...   N/2            ...   N
+-----------+-------+------------+-------+------------+
|BootSector |  ...  | BootSector |  ...  |            |
+-----------+-------+------------+-------+------------+
```

†In later versions this changed to LBN=n and not N/2.

Structure of the MFT

- The master file table contains the file record segments for all of the volume. The first 16 or so file record segments are reserved for special files. User file records start at file record #16.

```
   0   1   2   3   4   5   6   7   8   9    ...
+---+---+---+---+---+---+---+---+---+---+-----+
| M | M | L | V | A | R | B | B | B | Q |     |
| f | f | o | o | t | o | i | o | a | u |     |
| t | t | g | l | t | o | t | o | d | o |     |
|   | 2 | F | D | r | t | M | t | C | t | ... |
|   |   | i | a | D | D | a |   | l | a |     |
|   |   | l | s | e | i | p |   | u |   |     |
|   |   | e | d | f | r |   |   | s |   |     |
+---+---+---+---+---+---+---+---+---+---+-----+
```

Structure of a File Record

- Each file record is a fixed size and used to store meta data information for the file in a packed form where each tag starts with a [type, size] pair.
  - Name
  - Dates
  - Protection
  - Data streams (including size and retrieval information)
  - Indexes

64 bit system

- NTFS is designed for 64 bits
- Volume and file sizes are stored 64 bits
- NTFS and Windows NT in general also stored time as 64 bits with a 200 ns resolution starting at 1601

Not posted on the class web page

Attributes

- File data is stored in NTFS file records in what are called “attributes”
- Attributes are either resident or nonresident depending on the size of the data and room within the file record
- In a resident attribute the data is actually stored within the file record
- If the data is nonresident then the file records essentially contains [vcn, lcn, size] triples on where the data is actually stored on the disk

- Let’s look at an example

Not posted on the class web page

Compressed Files

- Data compression occurs on an individual attribute basis
  - Uses a patented compression format called LZNT1

- Allows for quick random read/write access to the file data

- On a write operation NTFS attempts to compress every 16 clusters if they result is less than 16 clusters then the compressed data is written to disk.

- On a read operation NTFS uncompresses and buffers (as a mapped file) every 16 clusters

Not posted on the class web page

Sparse Files

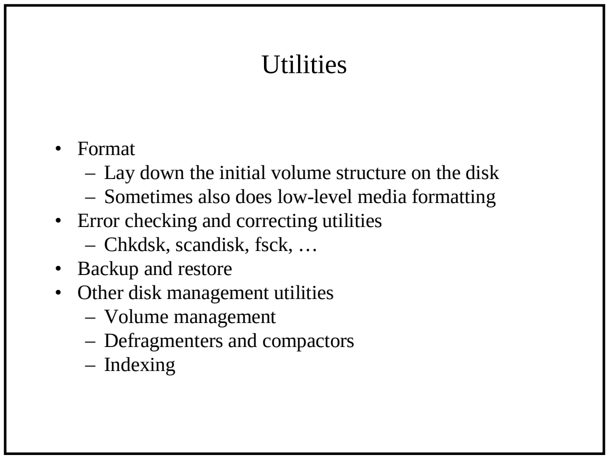
- Sparse files essentially follow the same paradigm as compressed files, but special case the situation where the data is all zeros

- The implication with compressed and sparse files is that the actual storage on the disk can be less than the actual files size
- Therefore it is possible to have a file larger than your disk
- And that writing into the middle of a file can fail with an out of space error

Not posted on the class web page

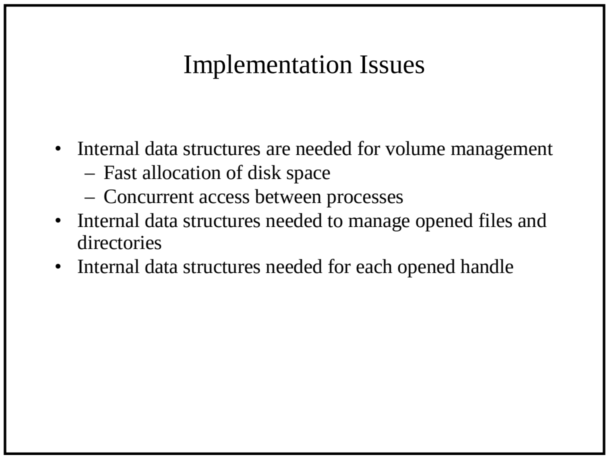
Consistency and Persistence

- Persistence of disk data is both a blessing and curse for any file system writer
- Data survives between reboots
  - “We saved your data before we crashed.”
- So does any data corruption
  - “Here’s your data back. A few bits got altered but you don’t mind, do you?”
- Need to guard the disk against hardware failures, software bugs, and idiots



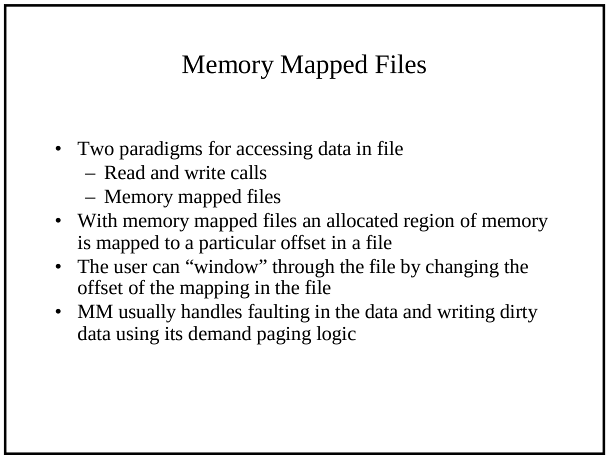
Utilities

- Format
  - Lay down the initial volume structure on the disk
  - Sometimes also does low-level media formatting
- Error checking and correcting utilities
  - Chkdsk, scandisk, fsck, …
- Backup and restore
- Other disk management utilities
  - Volume management
  - Defragmenters and compactors
  - Indexing



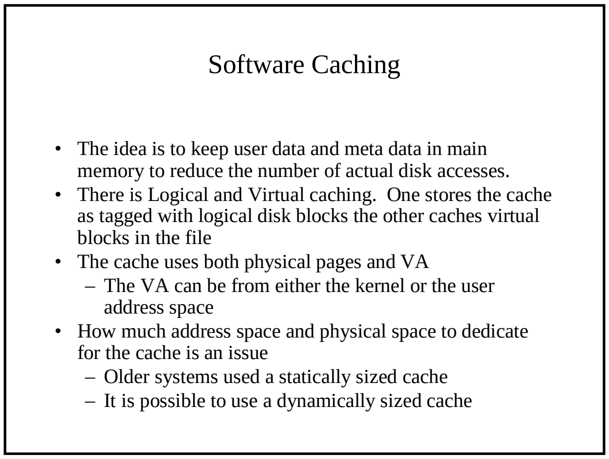
Implementation Issues

- Internal data structures are needed for volume management
  - Fast allocation of disk space
  - Concurrent access between processes
- Internal data structures needed to manage opened files and directories
- Internal data structures needed for each opened handle



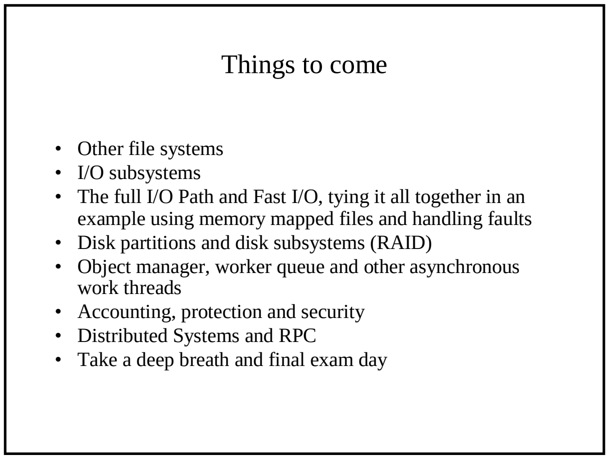
Memory Mapped Files

- Two paradigms for accessing data in file
  - Read and write calls
  - Memory mapped files
- With memory mapped files an allocated region of memory is mapped to a particular offset in a file
- The user can “window” through the file by changing the offset of the mapping in the file
- MM usually handles faulting in the data and writing dirty data using its demand paging logic



Software Caching

- The idea is to keep user data and meta data in main memory to reduce the number of actual disk accesses.
- There is Logical and Virtual caching. One stores the cache as tagged with logical disk blocks the other caches virtual blocks in the file
- The cache uses both physical pages and VA
  - The VA can be from either the kernel or the user address space
- How much address space and physical space to dedicate for the cache is an issue
  - Older systems used a statically sized cache
  - It is possible to use a dynamically sized cache



Things to come

- Other file systems
- I/O subsystems
- The full I/O Path and Fast I/O, tying it all together in an example using memory mapped files and handling faults
- Disk partitions and disk subsystems (RAID)
- Object manager, worker queue and other asynchronous work threads
- Accounting, protection and security
- Distributed Systems and RPC
- Take a deep breath and final exam day