COS221 - L25 - File Organisation (Part 2)

Linda Marshall

4 May 2023

Previously in a land far-far away....

- Physical level - how are our relations stored?
- Secondary media
- Storing fixed-length vs variable length records
- Placing records on disk
  - Record blocking - unspanned vs spanned
  - Allocating blocks on disk
  - File headers

Methods for Organising Records on the Disks

- **Sequential methods** include ordered and unordered record placement in files.
- **Hashing techniques** provides fast access to records under certain conditions. Hashing is used on internal search structures, external structures on disk and for dynamically growing files.

Methods for Organising Records on the Disks - Sequential methods

Unordered files (Heap)

- Records are placed in the order in which they are inserted. New records are placed at the end of the file.
- This method is often used with additional access paths - e.g. secondary indexes - or to collect and store records for future use.

Methods for Organising Records on the Disks - Sequential methods

Unordered files (Heap)

- *Insertion* of a record is efficient - the last disk block is copied into the buffer, the record added and the updated block rewritten to disk.
- *Searching* for a record involves a linear search.
- *Deletion* requires the block to be found, loaded into the buffer, remove the record from the buffer copy of the block and rewrite the block to disk.
  - Deleting large numbers of records results in wasted storage.
  - To facilitate reorganisation of the blocks and their records -
    - a deletion marker is placed in each record.
    - a new record can be placed in a large enough gap, which requires bookkeeping to track the gaps.

Methods for Organising Records on the Disks - Sequential methods

Ordered files

- Records of a file can be ordered by using an ordering field (typically a key).
- Files with ordered records are blocked and stored on contiguous cylinders to minimise seek time.
- Finding a record means finding a block which contains the record using a binary search.

Methods for Organising Records on the Disks - Sequential methods

Ordered files

- No advantage if searching for records based on the non-ordering field.
- Insertion and deletion are expensive. Techniques exist to limit the impact of the operations.
  - Keeping an overflow file for insertion.
  - This overflow is periodically sorted and merged into the master file during reorganisation.
- Modifying a field in a record depends on:
  - The search condition to locate the record and the field to be modified.
  - If the search condition involves the key, a binary search is done, otherwise a linear search takes place.
  - If fixed length records are stored, the field can be modified by writing directly to the same physical location on disk.
  - Modifying the key, requires the location of the record to change on disk.

Methods for Organising Records on the Disks - Sequential methods

Summary of average access time in block accesses for a specific record in a file of *b* blocks

Table 17.2 Average Access Times for a File of *b* Blocks under Basic File Organizations

Type of Organization	Access/Search Method	Average Blocks to Access a Specific Record
Heap (unordered)	Sequential scan (linear search)	$b/2$
Ordered	Sequential scan	$b/2$
Ordered	Binary search	$\log_2 b$

Methods for Organising Records on the Disks - Hashing techniques

Hashing

- A hash file must have a single hash field on which an equality condition can be completed. When the hash field is a key field, it is referred to as the hash key.
- In hashing, a function h called the hash function, is applied to the hash field. The result of this function, yields an address where the record is stored.
- Hashing can be applied to internal files. Internal file hashing can be extended to apply for external files. A further extension for dynamically growing files can also be made.

Methods for Organising Records on the Disks - Hashing techniques

Internal hashing

- Is implemented using a hash table which comprises of an array of records indexed from 0 to $M-1$. There are M record slots whose addresses correspond to the array indexes. An example of a hash function is $h(K) = K\%M$. It is possible that the hash function results in clashes.

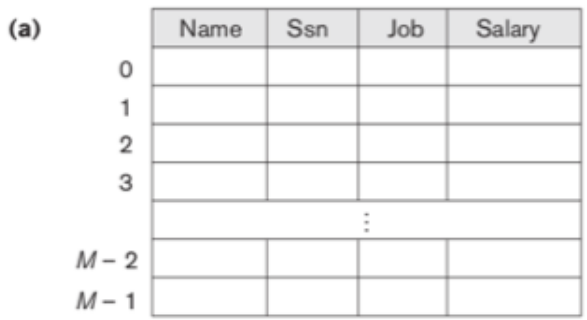


Figure 17.8
Internal hashing data structures. (a) Array of M positions for use in internal hashing. (b) Collision resolution by chaining records.

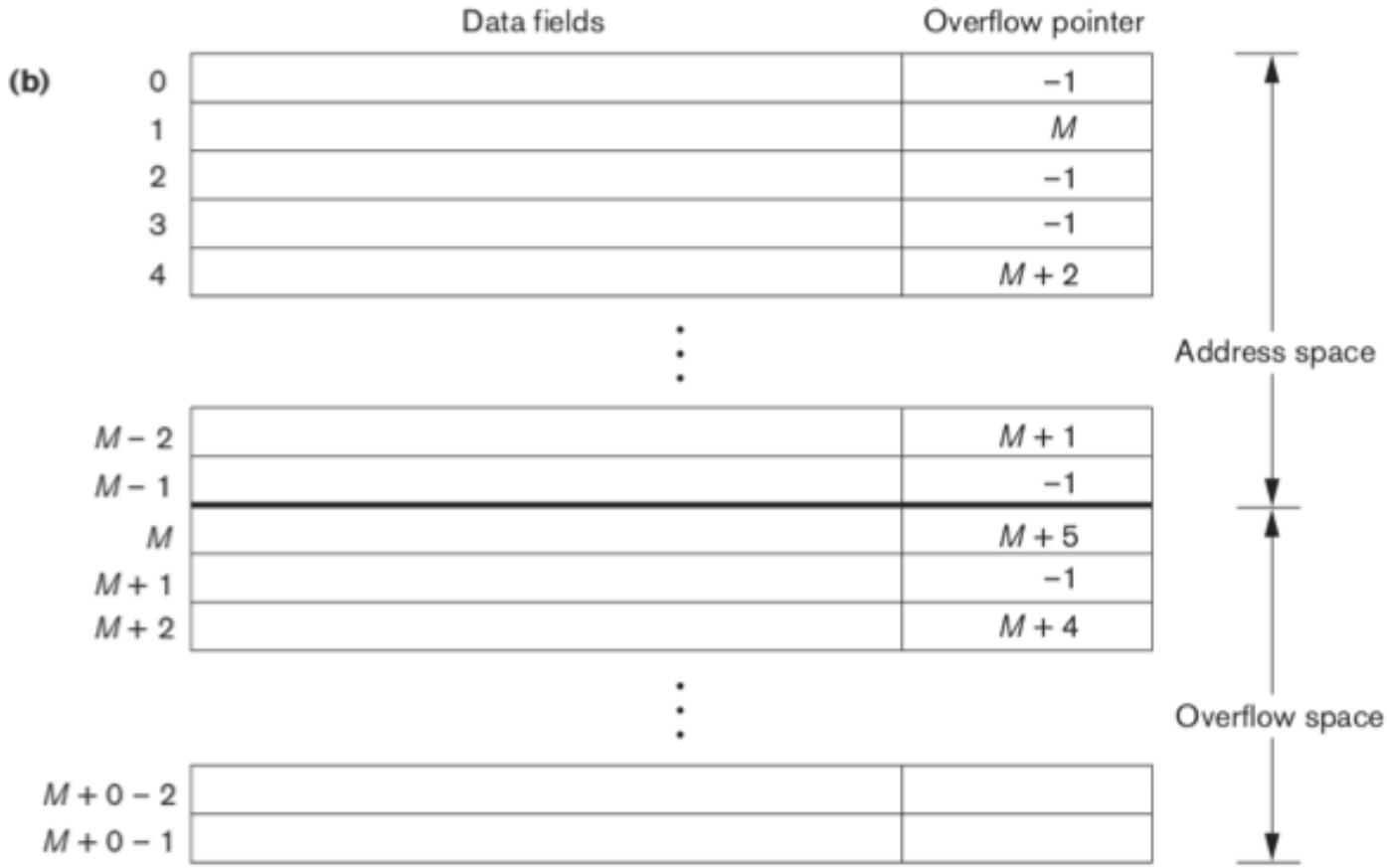
Methods for Organising Records on the Disks - Hashing techniques

Internal hashing

- There are 3 techniques to address clashes (collisions)
  - Open addressing - add the record after the address space in the next available position.
  - Chaining - Extend the array with a number of overflow positions (b) above. Add a pointer to the record which holds a pointer to where the record has been placed thereby maintaining a linked list for each overflow record.
  - Multiple hashing - apply a hash function again in case of a collision. When a 3rd collision occurs, resort to open addressing

Methods for Organising Records on the Disks - Hashing techniques

Internal hashing



Methods for Organising Records on the Disks - Hashing techniques

External hashing for disk files

- The target address space is made of buckets (either one block or a cluster of contiguous blocks). Each bucket holds multiple records.
- The hashing function maps a key into a relative bucket. A table, maintained by the file header converts the bucket number into the corresponding disk block address.

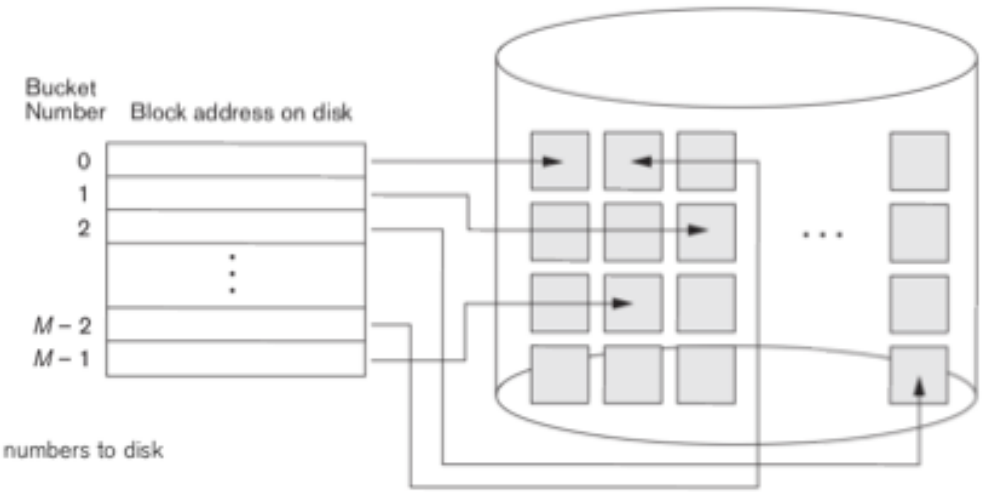


Figure 17.9
Matching bucket numbers to disk block addresses.

Methods for Organising Records on the Disks - Hashing techniques

External hashing for disk files

- If a fixed number of buckets is allocated, the hashing scheme is static. Each bucket can have an overflow scheme implemented.

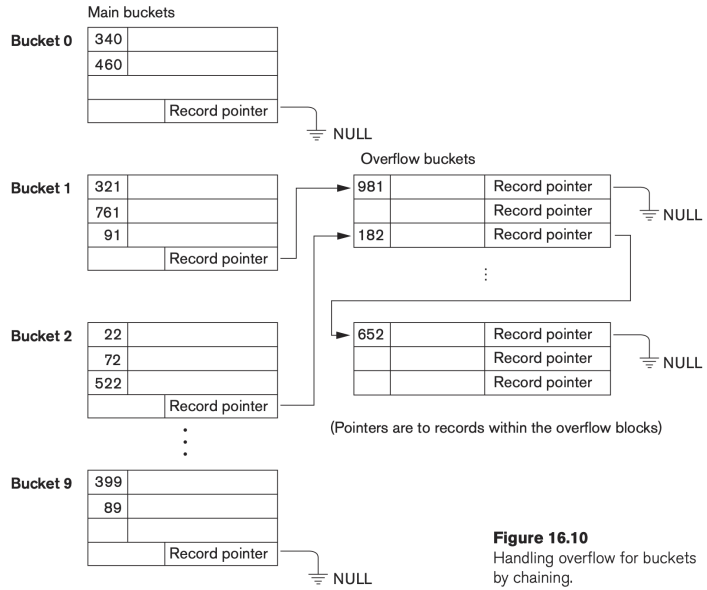


Figure 16.10
Handling overflow for buckets by chaining.

- Searching for a record on a value other than the hash field is expensive.
- Updating the hash field requires the record to be moved.

Methods for Organising Records on the Disks - Hashing techniques

Hashing allowing for dynamic file expansion

- A drawback of static hashing is the fixed address space.
- To solve the limitation of a fixed address space, dynamic hashing was proposed in 1978
- Dynamic hashing maintains a tree structure with:
  - internal nodes have a left pointer for a 0-bit in the hashed address and right pointer for a 1-bit
  - leaf nodes hold the pointer to the bucket with the records

Methods for Organising Records on the Disks - Hashing techniques

Dynamic hashing

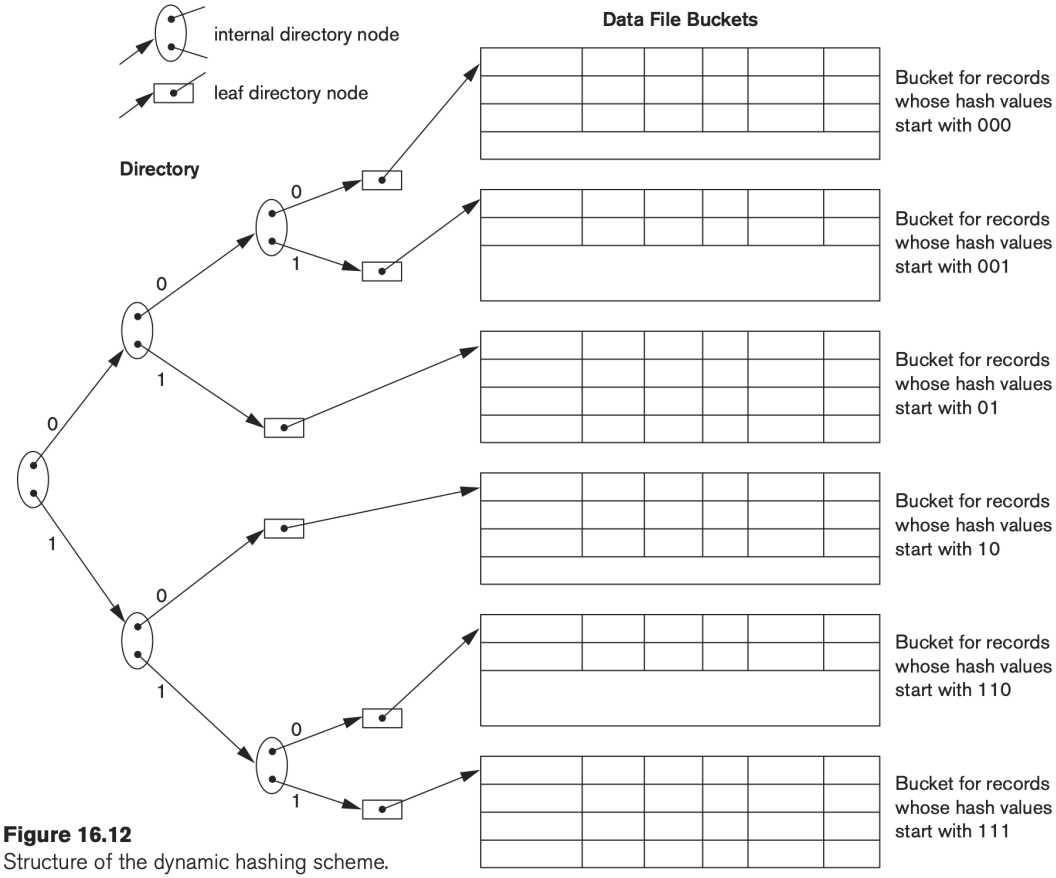


Figure 16.12
Structure of the dynamic hashing scheme.

Methods for Organising Records on the Disks - Hashing techniques

Dynamic hashing schemes to address the *address space problem*:
Extensible hashing

- Extensible hashing was proposed in 1979
- It has a more compact directory structure.
- An array of 2^d bucket addresses is kept, where d is the global depth of the dictionary.

Methods for Organising Records on the Disks - Hashing techniques

Extensible hashing

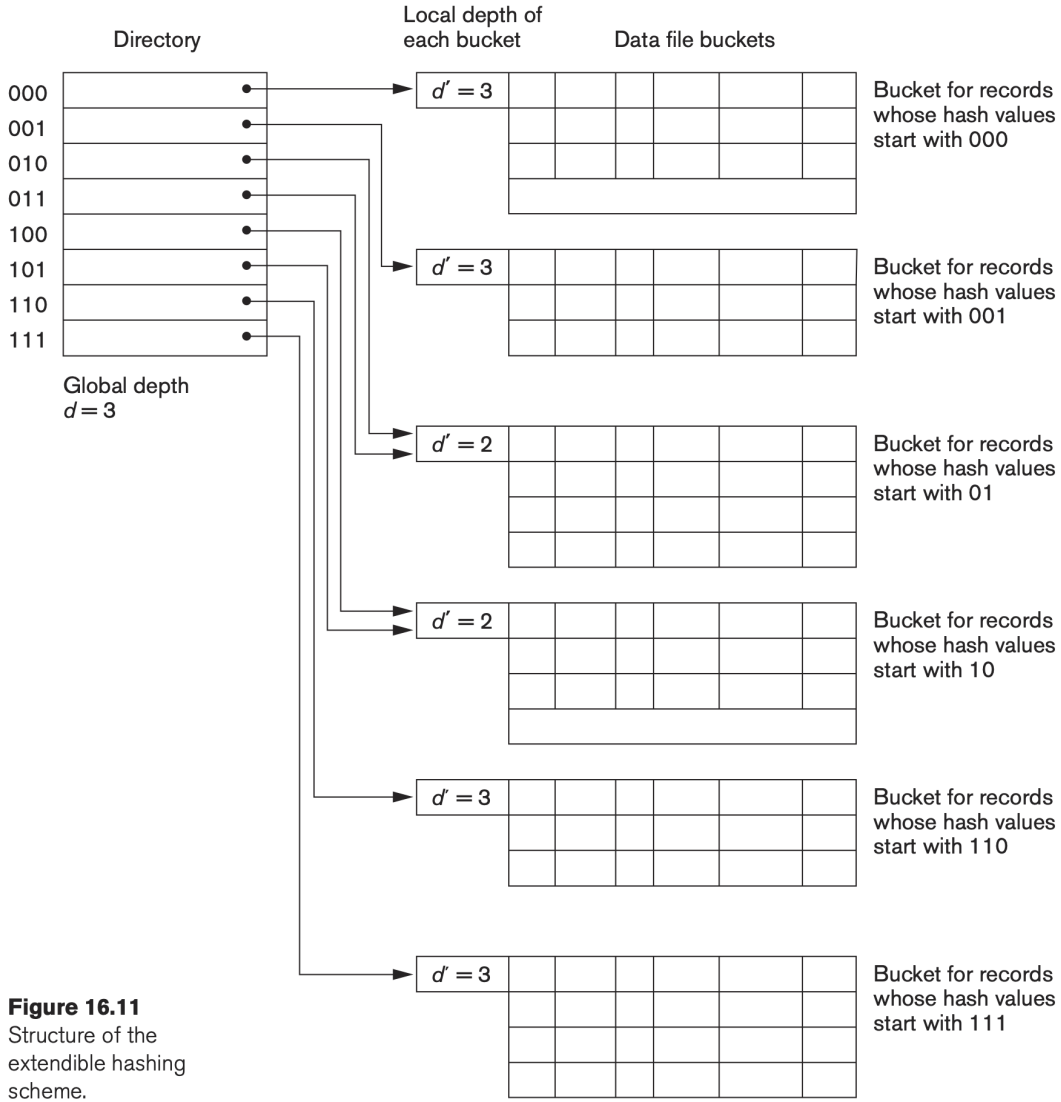


Figure 16.11
Structure of the extendible hashing scheme.

Methods for Organising Records on the Disks - Hashing techniques

Dynamic hashing schemes to address the *address space problem*:
Linear hashing

- Linear hashing was proposed in 1980.
- Linear hashing allows the buckets to shrink and expand buckets without the need for a dictionary.
- A collision results in a split in the buckets beginning at 0 and ending at $M - 1$.
- When all buckets have split, there will be $2M$ buckets.
- Splitting/joining of the buckets is further controlled by a load factor.
- The hash function updates when a split occurs. Ultimately, the hash function update propagates through the system.

Other File Organisations

- The assumption so far has been that records in the a file are of the same record type. Connecting fields represent the relationships between record types and therefore files. Relationships are therefore implemented by logical field references.
- B-trees are used in primary file organisation and will be discussed under indexing.

Parallelising and RAID Technology

- Redundant Array of Inexpensive (as it was, but Independent as it is now) Discs - RAID.
- Memory capacity has quadrupled every 2/3 years, disk capacity doubles every year, but disk access times have increased by less than 10% each year.
- RAID therefore attempts to address and mismatch in different rates of performance.
- The solution, an array of independent disks acting as a single logical disk. Data is distributed across multiple disks making it appear as if it is being written to one (called data stripping).

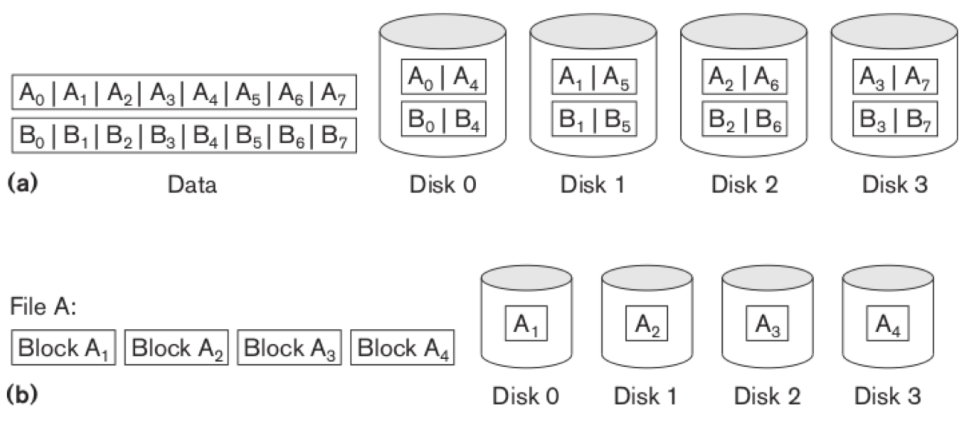


Figure 17.13
Striping of data across multiple disks. (a) Bit-level striping across four disks. (b) Block-level striping across four disks.

Parallelising and RAID Technology

Improving RAID

- reliability: reliability of disks is measured in Mean Time Between Failures (MTBF) and can be anything from days to years. The less reliable the disk, the more important it is to ensure data integrity. To do this, either:
  - redundancy is introduced by mirroring/shadowing - data is written to two identical physical disks. Data is usually read from the disk that performs better. If it fails, the reading of data moves to the other disk until the failed one has been repaired; or
  - information is stored that can be used to reconstruct the failed data.
- performance: data stripping is used to enhance performance. This can take place on either the bit-level or the block-level.

Parallelising and RAID Technology

RAID Organisation and levels is defined based on the granularity of the stripping and the pattern used to compute redundant information. Initial levels from 1 to 5 where defined. Later levels 0 and 6 were added.

- RAID level 0 - uses data stripping, has no redundant data
- RAID level 1 - uses mirrored disks and reads from the one with the shortest seek time
- RAID level 2 - uses Hamming code which contain parity bits
- RAID level 3 - uses single parity and relies on the disk controller to determine disk failure
- RAID level 4 - uses block-level stripping
- RAID level 5 - uses block-level stripping and distributes data and parity across all disks
- RAID level 6 - P + Q redundancy scheme applied

Parallelising and RAID Technology

- Easiest to rebuild on RAID level 1, mostly used for storing logs (critical applications).
- Levels 3 and 5 preferred for large volume storage with level 3 providing higher transfer rates.
- Most popular, level 0 with stripping, level 1 with mirroring and level 5 with an extra drive for parity

Other Storage Systems

- **Storage Area Networks** - with the growth of ERP, data warehouses are created storing aggregate data. SANs are online and are configured as nodes on high-speed network. SAN-attached devices appear as SCSI devices. They are however multiple RAID systems connected together. SANs are growing, but need to solve combining devices from multiple vendors.
- **Network Attached Storage** - are servers which do not only provide normal server functionality, but provide additional storage for file sharing. NAS devices reside anywhere on a LAN. A NAS box/head acts as the interface between the NAS system and the network clients. Provide wider operating system options. Usually use RAID levels 0, 1 and 5

Other Storage Systems

- **iSCSI (Internet SCSI) Storage Systems** - allow clients to send SCSI commands. Mostly used in small to medium sized businesses. Usually follows one of two approaches:
  - Fiber Channel over IP - transmits Fiber Channel control codes and data as IP packets
  - Fiber Channel over Ethernet - iSCSI without the IP. Easy for vendors to add their products.