## 4.11 FREE-SPACE MANAGEMENT

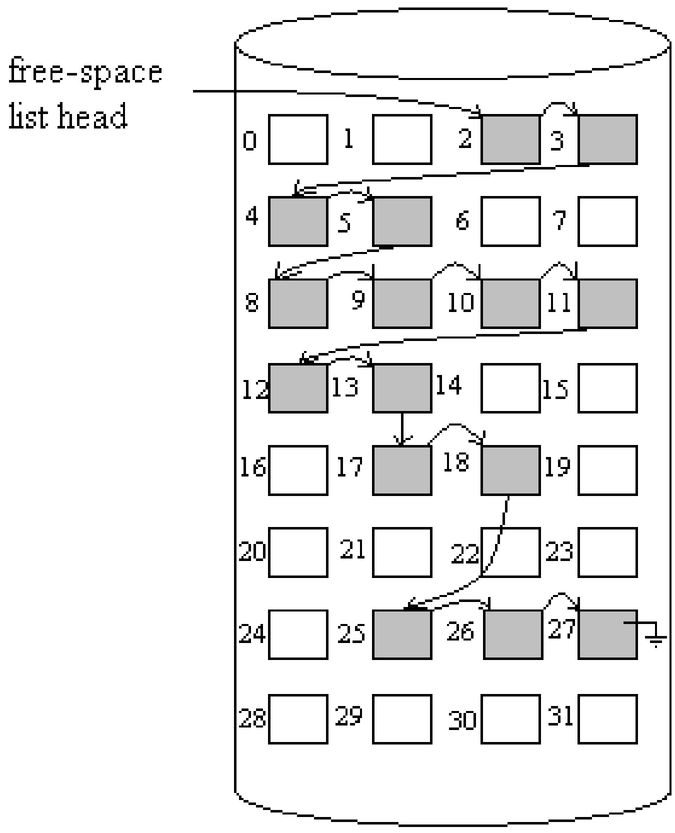
- Since disk space is limited, we need to reuse the space from deleted files for new files, if possible.
- To keep track of free disk space, the system maintains a free-space list.
- The free-space list records all free disk blocks – those not allocated to some file or directory.
- To create a file, we search the free-space list for the required amount of space, and allocate that space to the new file.
- This space is then removed from the free-space list.
- When a file is deleted, its disk space is added to the free-space list.

### 1. Bit Vector

- The free-space list is implemented as a bit map or bit vector.
- Each block is represented by 1 bit. If the block is free, the bit is 1; if the block is allocated, the bit is 0.
- For example, consider a disk where block 2,3,4,5,8 , and the rest of the block are allocated. The free space bit map would be 00111100100000 …
- The main **advantage** of this approach is it's simplicity and efficiency in finding the first free block, or n consecutive free blocks on the disk.

### 2. Linked List

- Another approach to free-space management is to link together all the free disk blocks, keeping a pointer to the first free block in a special location on the disk and caching it in memory.
- This first block contains a pointer to the next free disk block, and so on.
- In our example, we would keep a pointer to block 2, as the first free block. Block 2 would contain a pointer to block 3, which would point to block 4, which would point to block 5, which would point to block 8, and so on.
- However, this scheme is not efficient; to traverse the list, we must read each block, which requires substantial I/O time.



**3. Grouping**

- A modification of the free-list approach is to store the addresses of n free blocks in the first free block.
- The first n-1 of these blocks are actually free.
- The last block contains the addresses of another n free blocks, and so on.
- The importance of this implementation is that the addresses of a large number of free blocks can be found quickly.

**4. Counting**

- We can keep the address of the first free block and the number n of free contiguous blocks that follow the first block.
- Each entry in the free-space list then consists of a disk address and a count.
- Although each entry requires more space than would a simple disk address, the overall list will be shorter, as long as the count is generally greater than 1.

**5. Space Maps**

- Oracle's **ZFS** file system (found in Solaris and other operating systems) was designed to encompass huge numbers of files, directories, and even file systems.

- In its management of free space, ZFS uses a combination of techniques to control the size of data structures and minimize the I/O needed to manage those structures.
- First, ZFS creates **meta slabs** to divide the space on the device into chunks of manageable size. A given volume may contain hundreds of meta slabs. Each meta slab has an associated space map.
- ZFS uses the counting algorithm to store information about free blocks. Rather than write counting structures to disk, it uses log-structured file-system techniques to record them.
- The space map is a log of all block activity (allocating and freeing), in time order, in counting format.
- When ZFS decides to allocate or free space from a meta slab, it loads the associated space map into memory in a balanced-tree structure (for very efficient operation), indexed by offset, and replays the log into that structure.
- The in-memory space map is then an accurate representation of the allocated and free space in the meta slab. ZFS also condenses the map as much as possible by combining contiguous free blocks into a single entry.
- Finally, the free-space list is updated on disk as part of the transaction-oriented operations of ZFS.