Free Block List

Keeps track of all unallocated blocks on disk

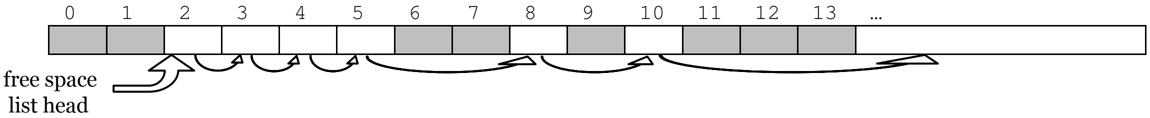
#1 Bit Vector

001111001111100011 …
1 bit for each disk block, 0 = unallocated, 1 = allocated

+ Simple
- Expensive to scan through
- Finding contiguous blocks is expensive
- Hard to keep in sync with disk

    1 TB disk -> 256 Mb = 32 MB (bit vector size)

#2 Linked List



+ Low on disk overhead
+ Simple
- Formatting is expensive (writing a pointer in every block)
- Expensive to allocate multiple blocks

#3 Linked List with Indexed Allocation

2 types of pointer:
(1) pointers to free block
(2) pointers to free block index

An index block contains pointer to free blocks and pointers to other index blocks

Reliability for File Systems

Scenario: Creating a file

- Directory entry update
- Free block list update
- File data block
- File control block

#1 Proceed in the following order:

1. Write data block
2. Write file control block
3. Write directory entry
4. Update free block list

Doesn't work

- Disk controller reorder writes
- Force such order would results in poor disk performance

#2 Consistency check:

- Walk over the entire file system, check consistency of metadata
- Check on boot
- Unix: fsch tool
- Clean shutdown flag, disk flush/consistent flag

#3 Log-structured file system
Intuition:

- Keep a log of all writes
- For each entry (write), store "old" value and "new" value

First, write transactions to log
Then, perform actual writes on disk
Once synched to disk, remove log entries

After a crash:
Inspect state of log:

- Entries that are complete
- Entries that are incomplete

Log is a separate region on disk, writing to log is fast compare to actual data writes

Performance and Efficiency

#1 Efficiency

- Storage usage (user data vs. overhead, fragmentation, etc.)
- Example in Unix
  - Inodes are pre-allocated
    - Scattered over the disk
    - Allow for contiguous allocation of disk blocks