Bucknell UNIVERSITY

File Systems Implementation

CSCI 315 Operating Systems Design
Department of Computer Science

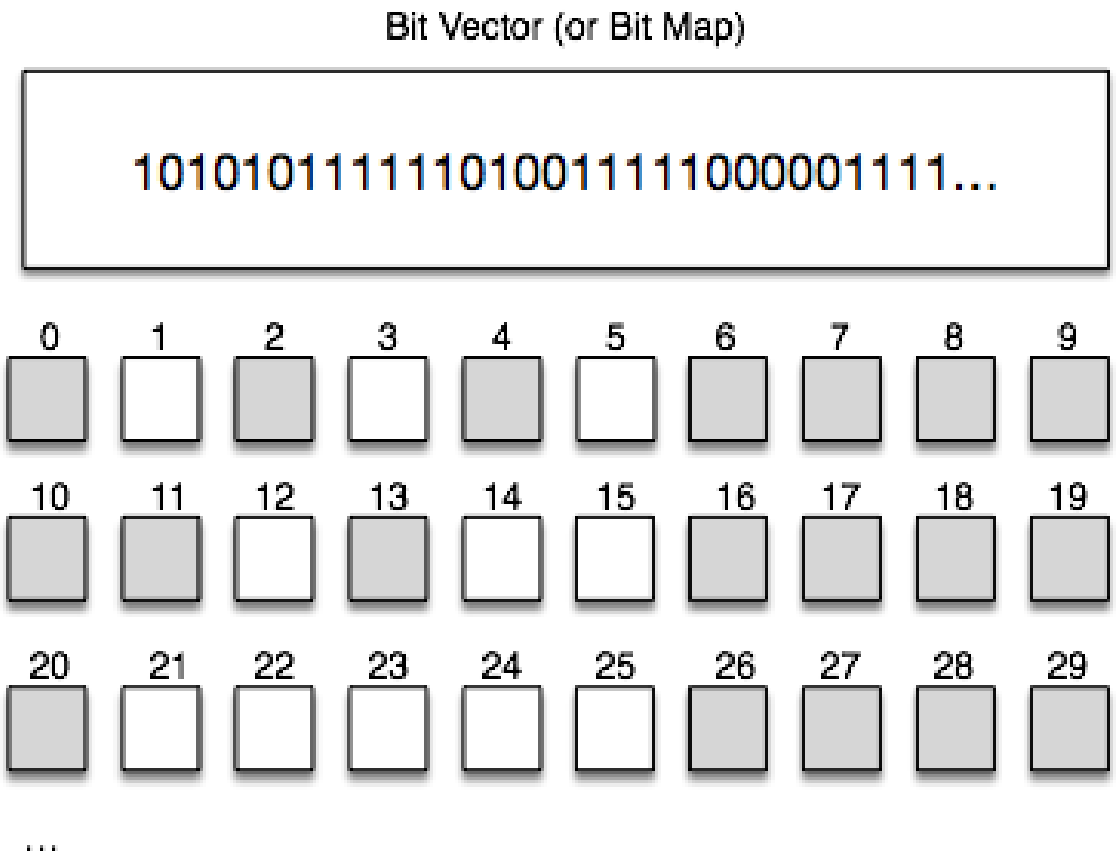
Based on Slides From Felipe Perrone

Free-Space Management

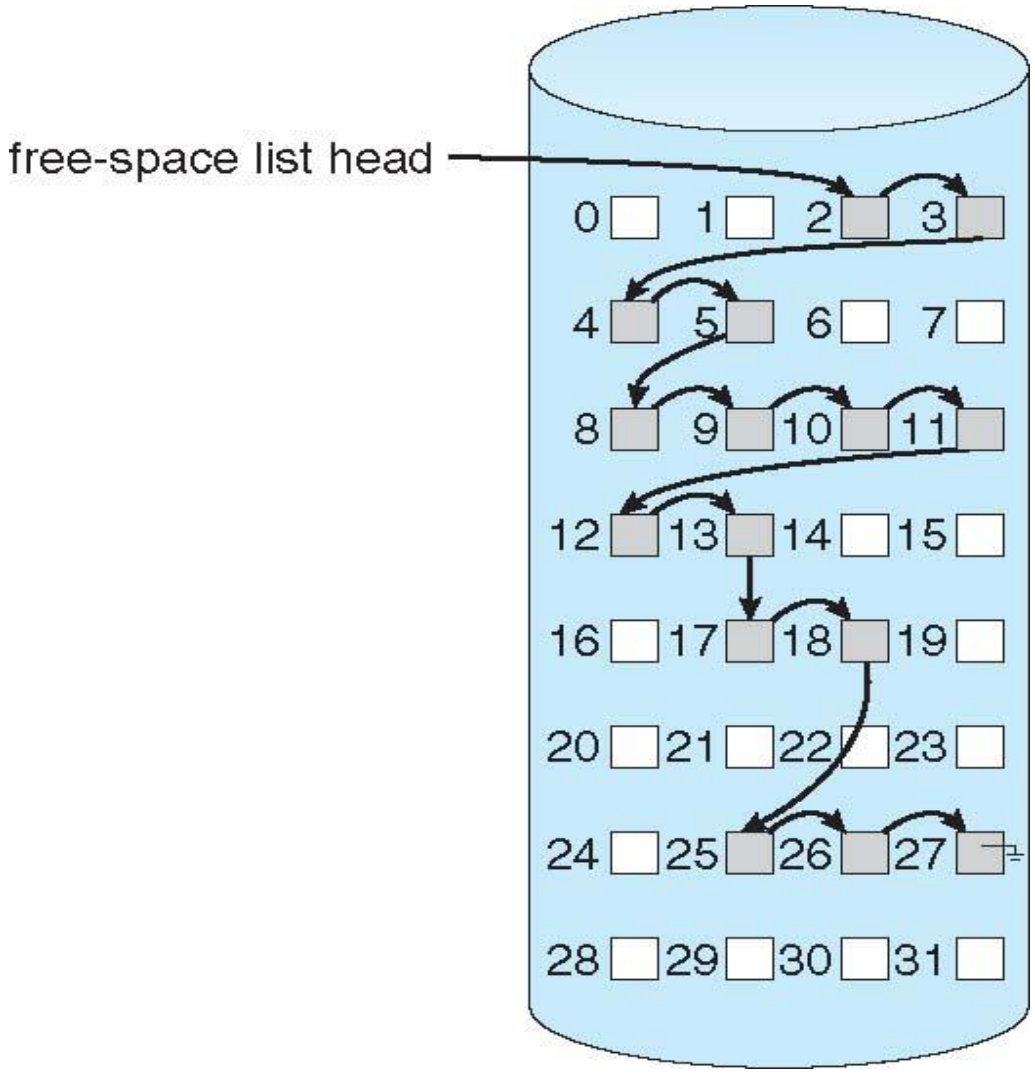
- Bit vector (1 bit per disk block)
- Linked list (free list)
- Grouping
  - like linked list: first free block has n block addresses (the n-1 addresses are free blocks, the nth is the address of a block with the next bundle of addresses)
- Counting
  - like linked list, but each node points to a cluster of contiguous, free blocks

The OS can cache in memory the free-space management structures for increased performance. Depending on disk size, this may not be easy.

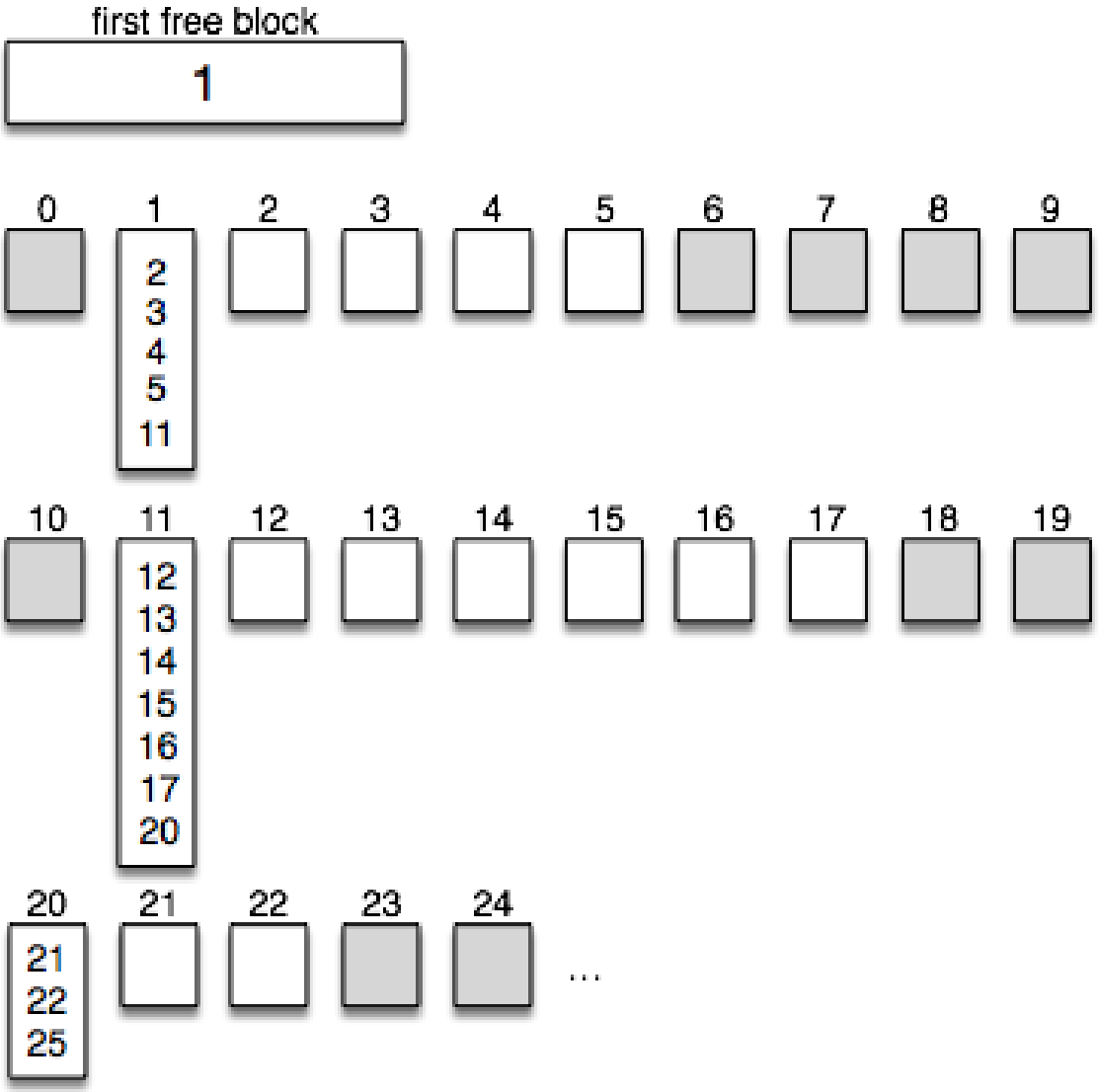
Bit Vector


Bit Vector (or Bit Map)
1010101111110100111110000011111...
0 1 2 3 4 5 6 7 8 9
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29
...

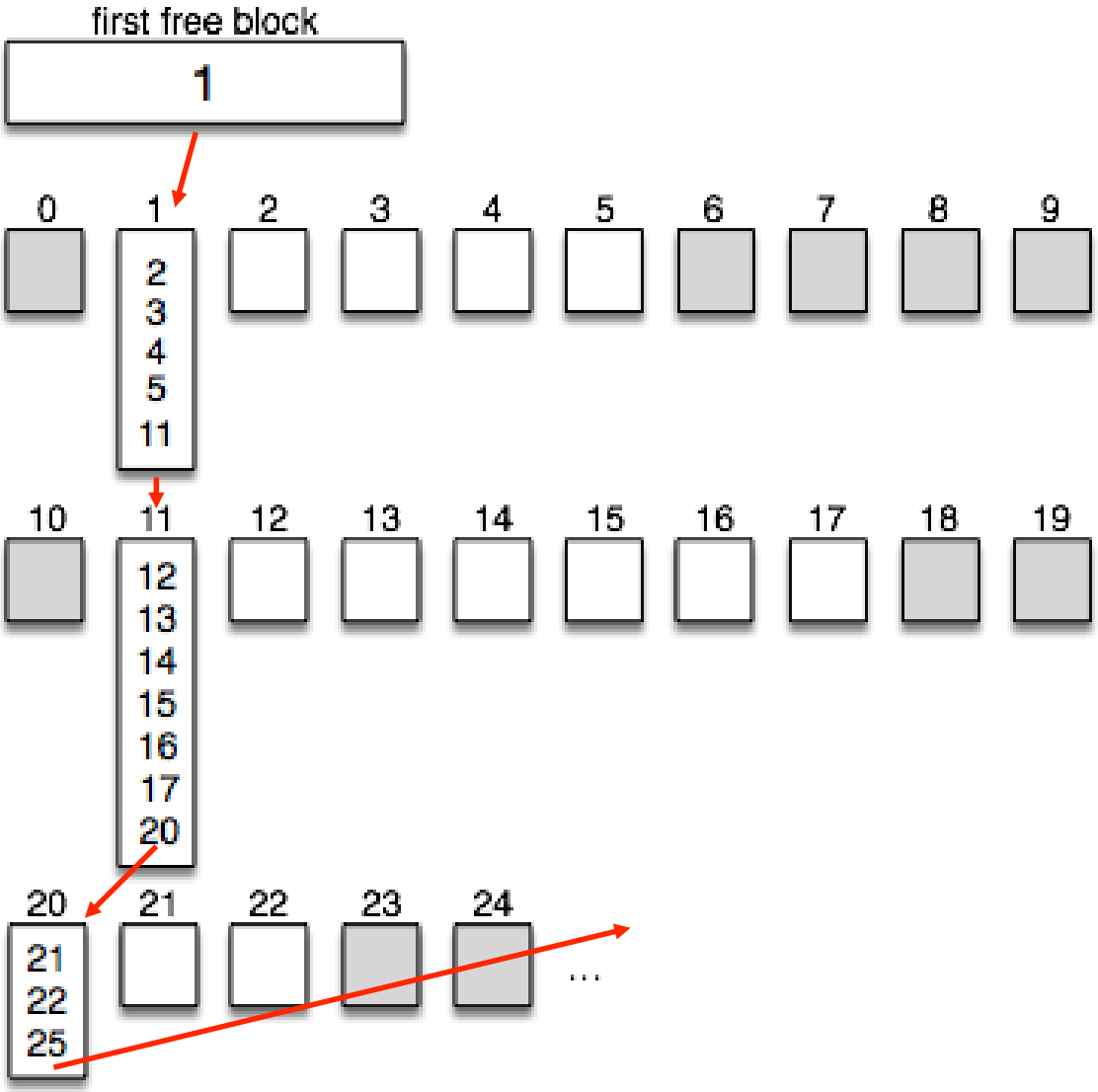
Linked List


free-space list head
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

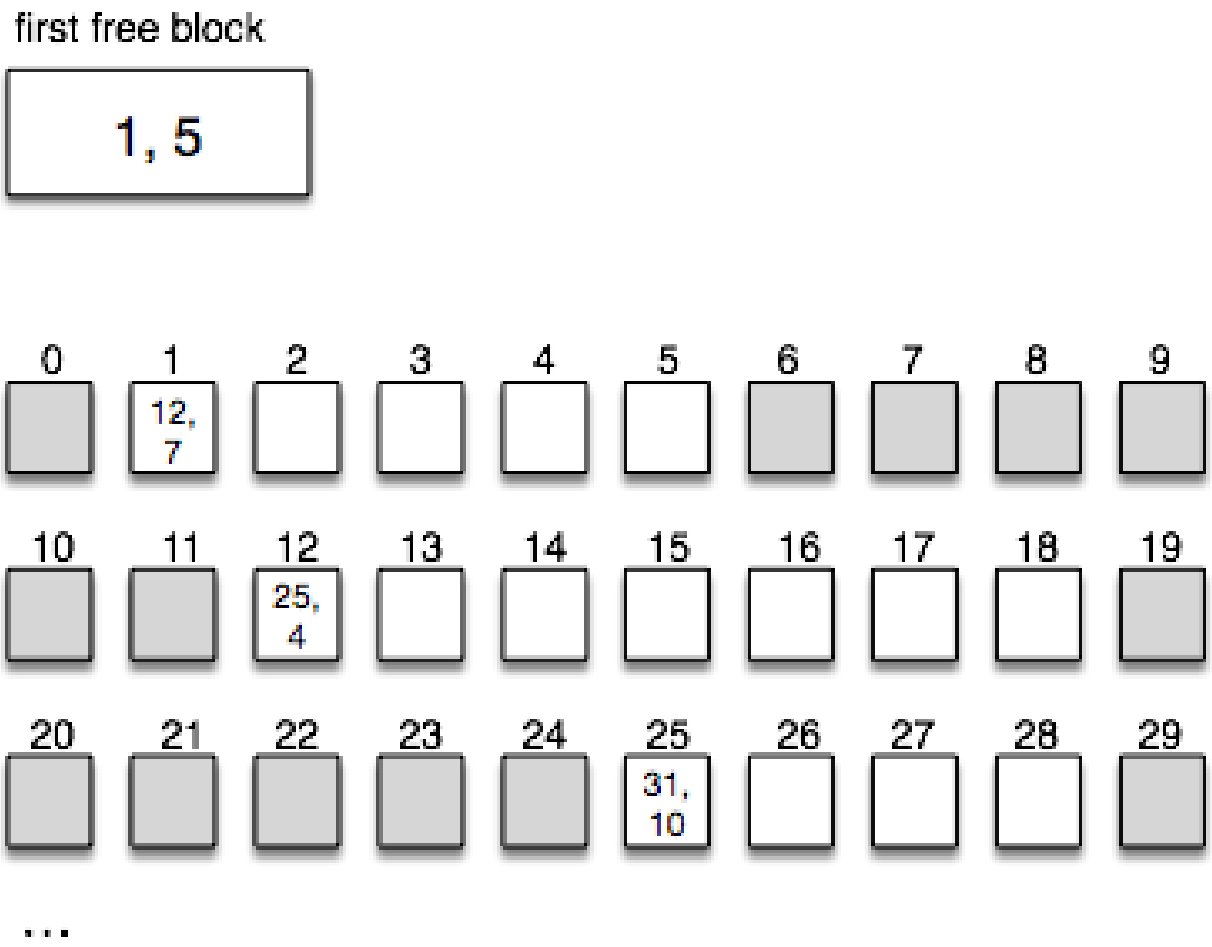
Grouping


first free block
1
0
1
2
3
4
5
6
7
8
9
2
3
4
5
11
10
11
12
13
14
15
16
17
18
19
12
13
14
15
16
17
20
20
21
22
23
24
21
22
25
...

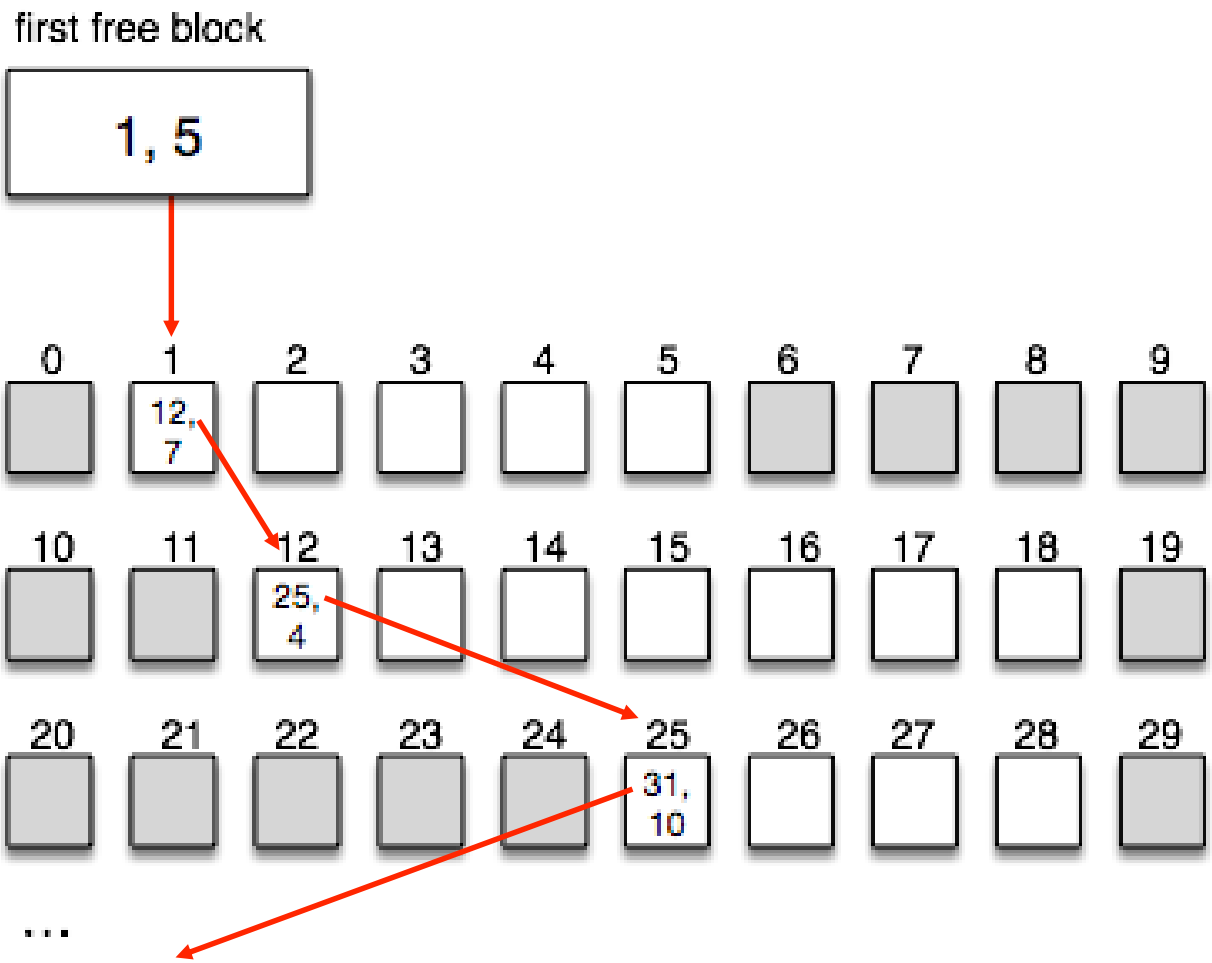
Grouping


first free block
1
0
1
2
3
4
5
6
7
8
9
2
3
4
5
11
10
11
12
13
14
15
16
17
18
19
12
13
14
15
16
17
20
20
21
22
23
24
21
22
25
...

Counting


first free block
1, 5
0
1
12, 7
2
3
4
5
6
7
8
9
10
11
12
25, 4
13
14
15
16
17
18
19
20
21
22
23
24
25
31, 10
26
27
28
29
...

Counting


first free block
1, 5
0
1
2
3
4
5
6
7
8
9
12,
7
10
11
12
13
14
15
16
17
18
19
25,
4
20
21
22
23
24
25
26
27
28
29
31,
10
...

Recovery

- Consistency checking – compares data in directory structure with data blocks on disk, and tries to fix inconsistencies.
- Use system programs to *back up* data from disk to another storage device (floppy disk, magnetic tape).
- Recover lost file or disk by *restoring* data from backup.

Log Structured File Systems

- Log structured (or journaling) file systems record each update to the file system as a transaction.
- All transactions are written to a log. A transaction is considered committed once it is written to the log. However, the file system may not yet be updated.
- The transactions in the log are asynchronously written to the file system. When the file system is modified, the transaction is removed from the log.
- If the file system crashes, all remaining transactions in the log must still be performed.

Efficiency and Performance

- Efficiency dependent on:
  - Disk allocation and directory algorithms
  - Types of data kept in file's directory entry
  - Pre-allocation or as-needed allocation of metadata structures

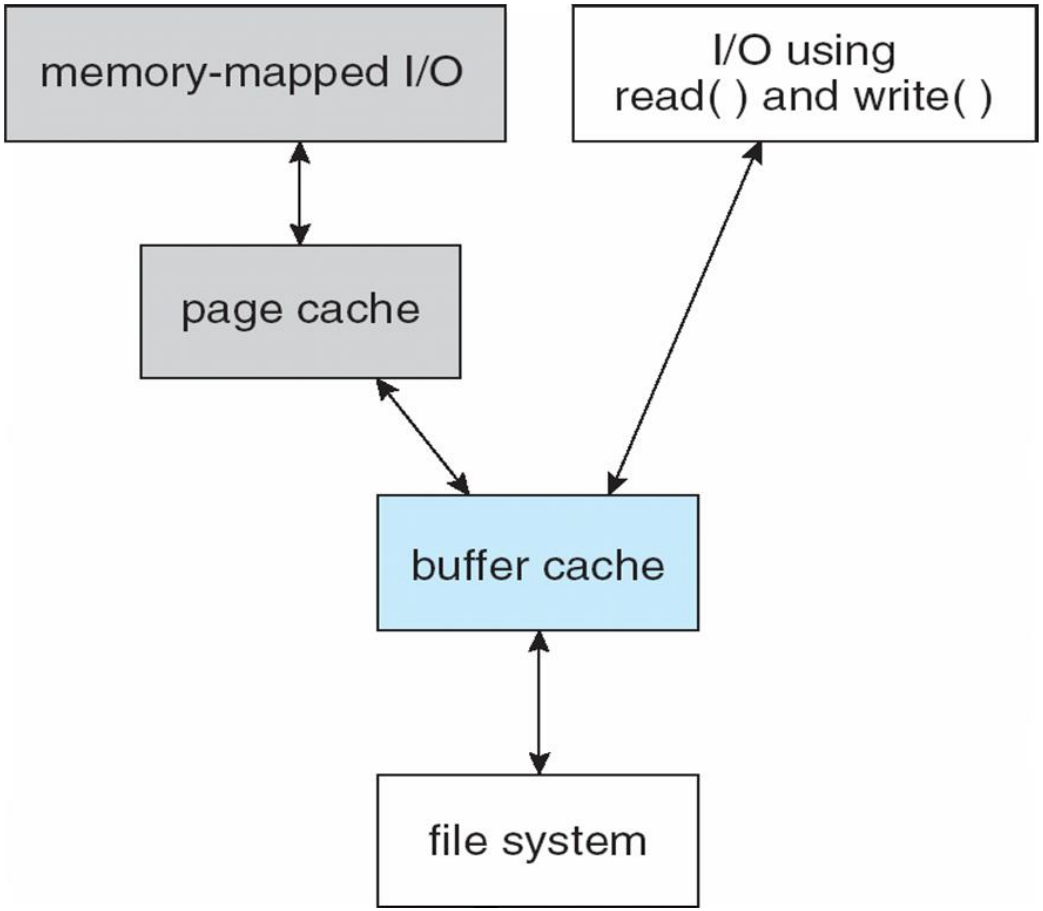
Efficiency and Performance (Cont.)

- Performance
  - Keeping data and metadata close together
  - **Buffer cache** – separate section of main memory for frequently used blocks
  - **Synchronous** writes sometimes requested by apps or needed by OS
    - No buffering / caching – writes must hit disk before acknowledgement
  - **Asynchronous** writes more common, buffer-able, faster
  - **Free-behind** and **read-ahead** – techniques to optimize sequential access
  - Reads frequently slower than writes

Page Cache

- A **page cache** caches pages rather than disk blocks using virtual memory techniques and addresses
- Memory-mapped I/O uses a page cache
- Routine I/O through the file system uses the buffer (disk) cache
- This leads to the following figure

I/O Without a Unified Buffer Cache

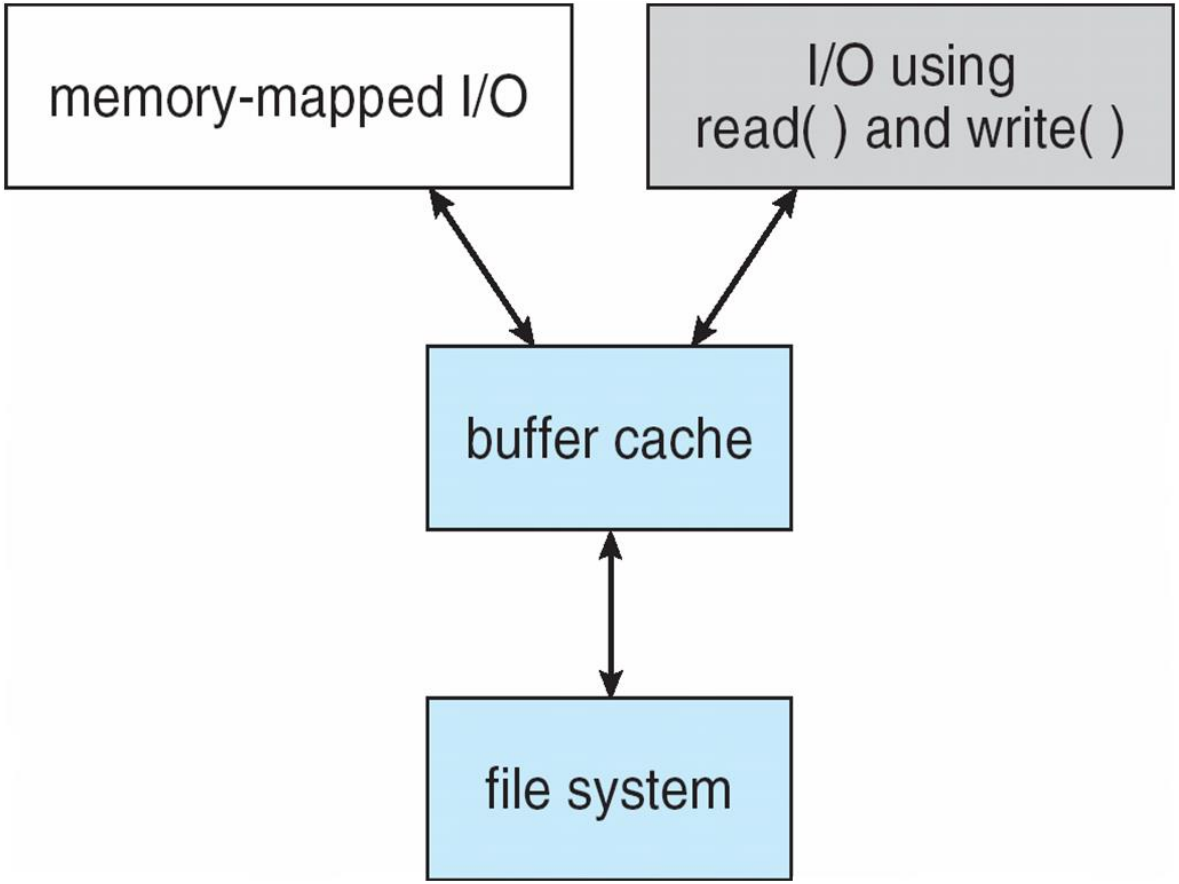


Unified Buffer Cache

- A **unified buffer cache** uses the same page cache to cache both memory-mapped pages and ordinary file system I/O to avoid **double caching**

- But which caches get priority, and what replacement algorithms to use?

I/O Using a Unified Buffer Cache



Recovery

- **Consistency checking** – compares data in directory structure with data blocks on disk, and tries to fix inconsistencies
  - Can be slow and sometimes fails
- Use system programs to **back up** data from disk to another storage device (magnetic tape, other magnetic disk, optical)
- Recover lost file or disk by **restoring** data from backup

Log Structured File Systems

- **Log structured** (or **journaling**) file systems record each metadata update to the file system as a **transaction**
- All transactions are written to a log
  - A transaction is considered committed once it is written to the log (sequentially)
  - Sometimes to a separate device or section of disk
  - However, the file system may not yet be updated
- The transactions in the log are asynchronously written to the file system structures
  - When the file system structures are modified, the transaction is removed from the log
- If the file system crashes, all remaining transactions in the log must still be performed
- Faster recovery from crash, removes chance of inconsistency of metadata