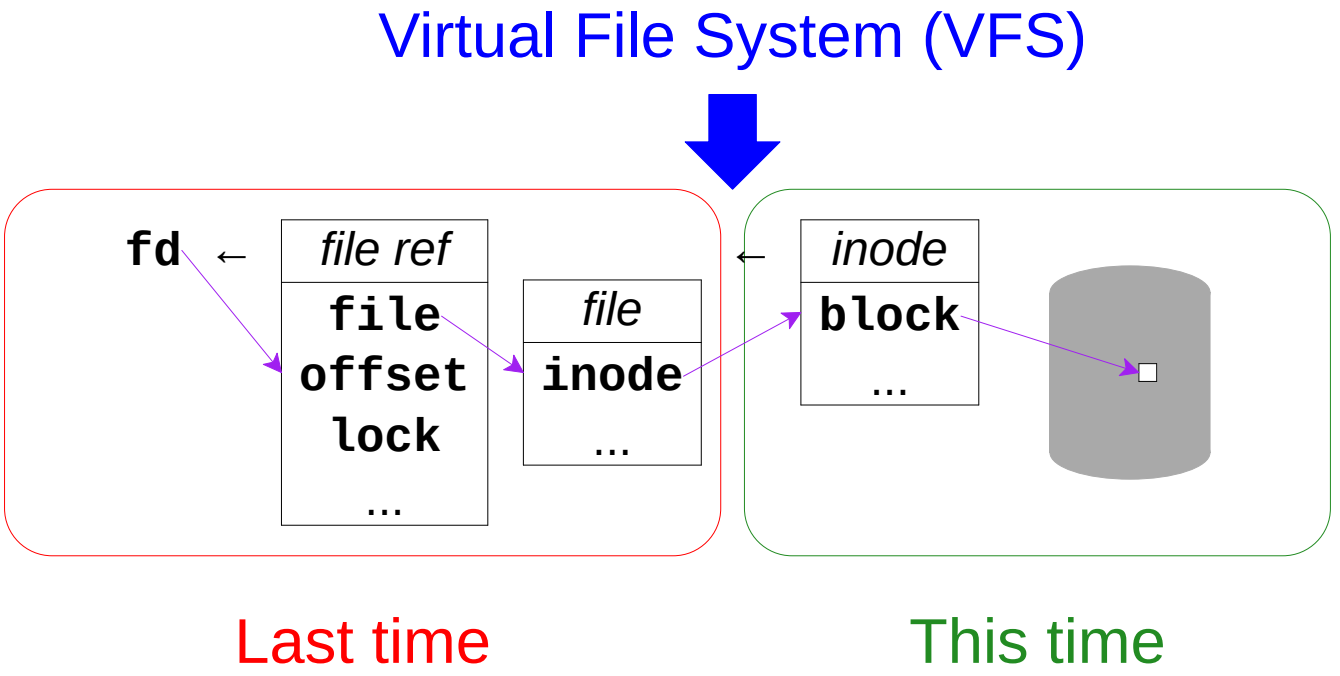
File System Implementation



Implementation Challenges

- Finding a file by path name *quickly*
- Finding a file's content *quickly*
- Finding empty space *quickly*
- Recovering from failure

Finding a File

- Each directory is like a file
  - Linear list of content
  - Sorted tree for content
  - Hash table for content

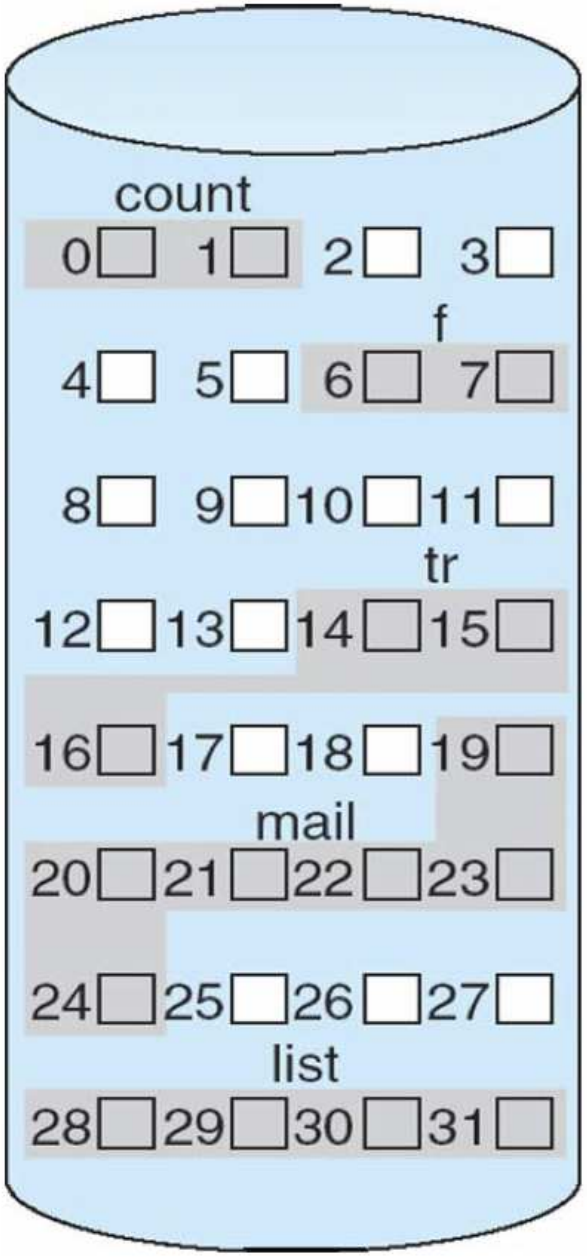
  with all the usual trade-offs

Finding a File's Data

Three popular choices:

- Contiguous allocation
- Linked allocation
- Indexed allocation

Contiguous Allocation



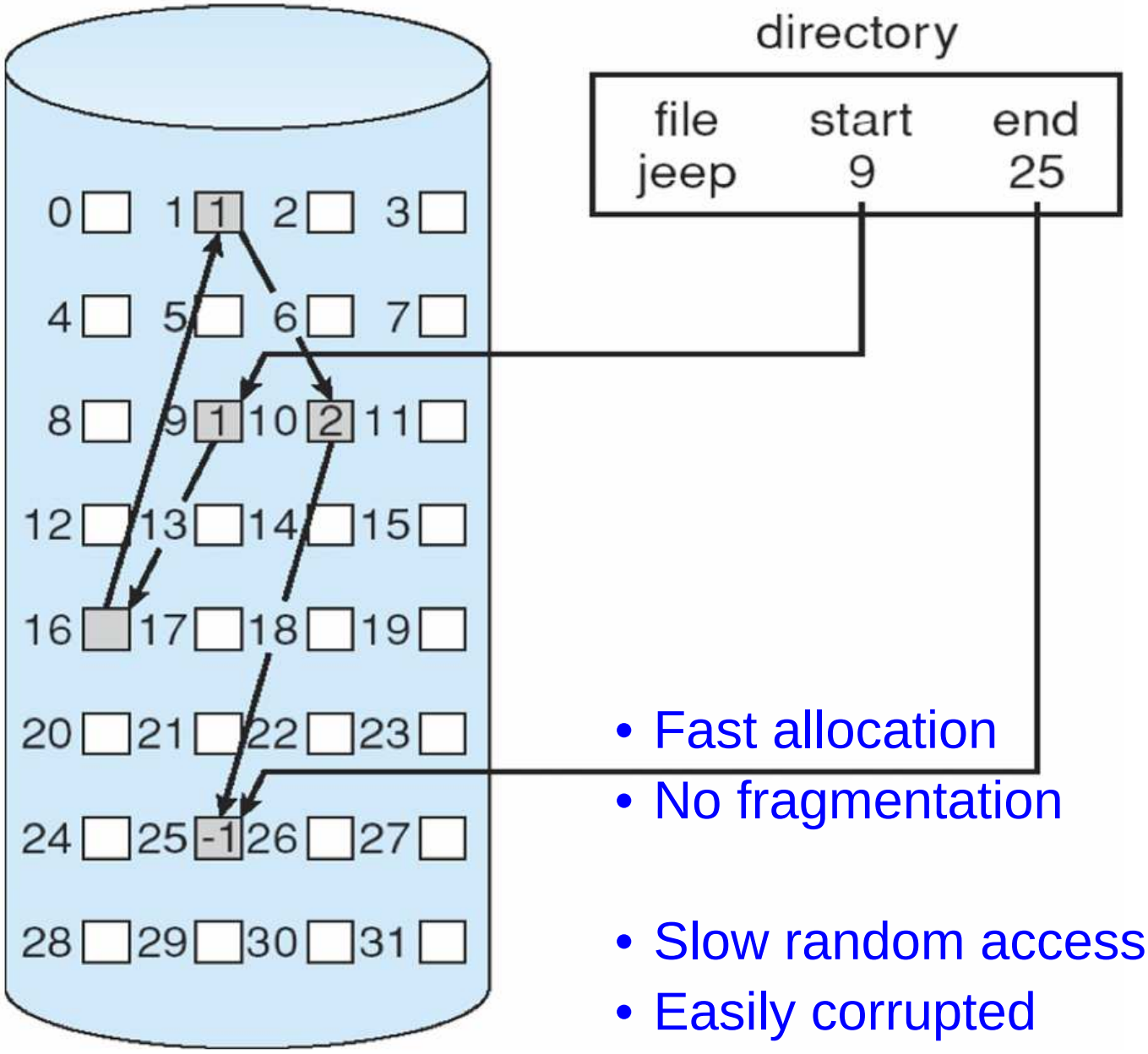
directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

- Simple tracking
- Fast random access

- Growing may require moving
- Fragmentation

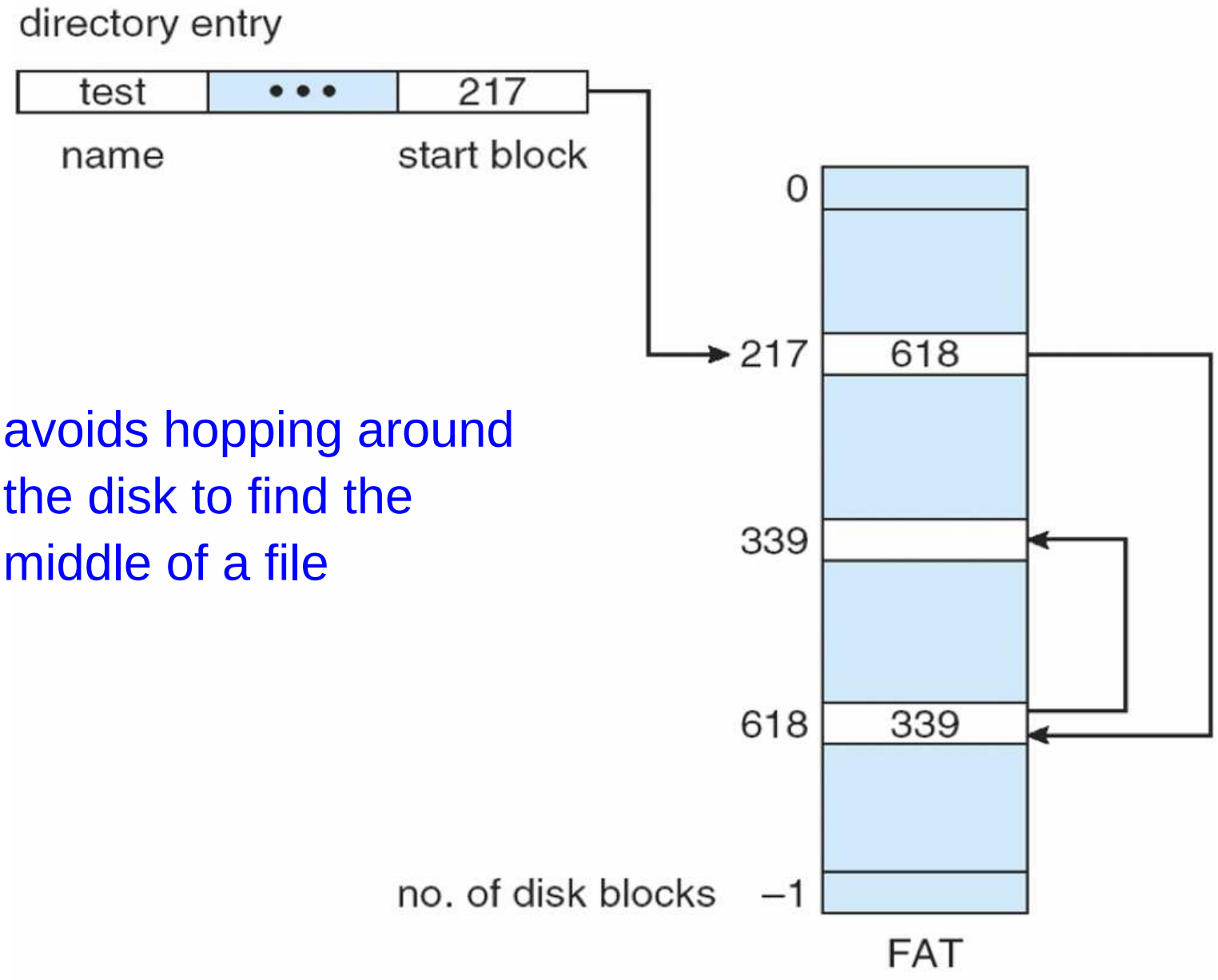
Linked Allocation



- Fast allocation
- No fragmentation

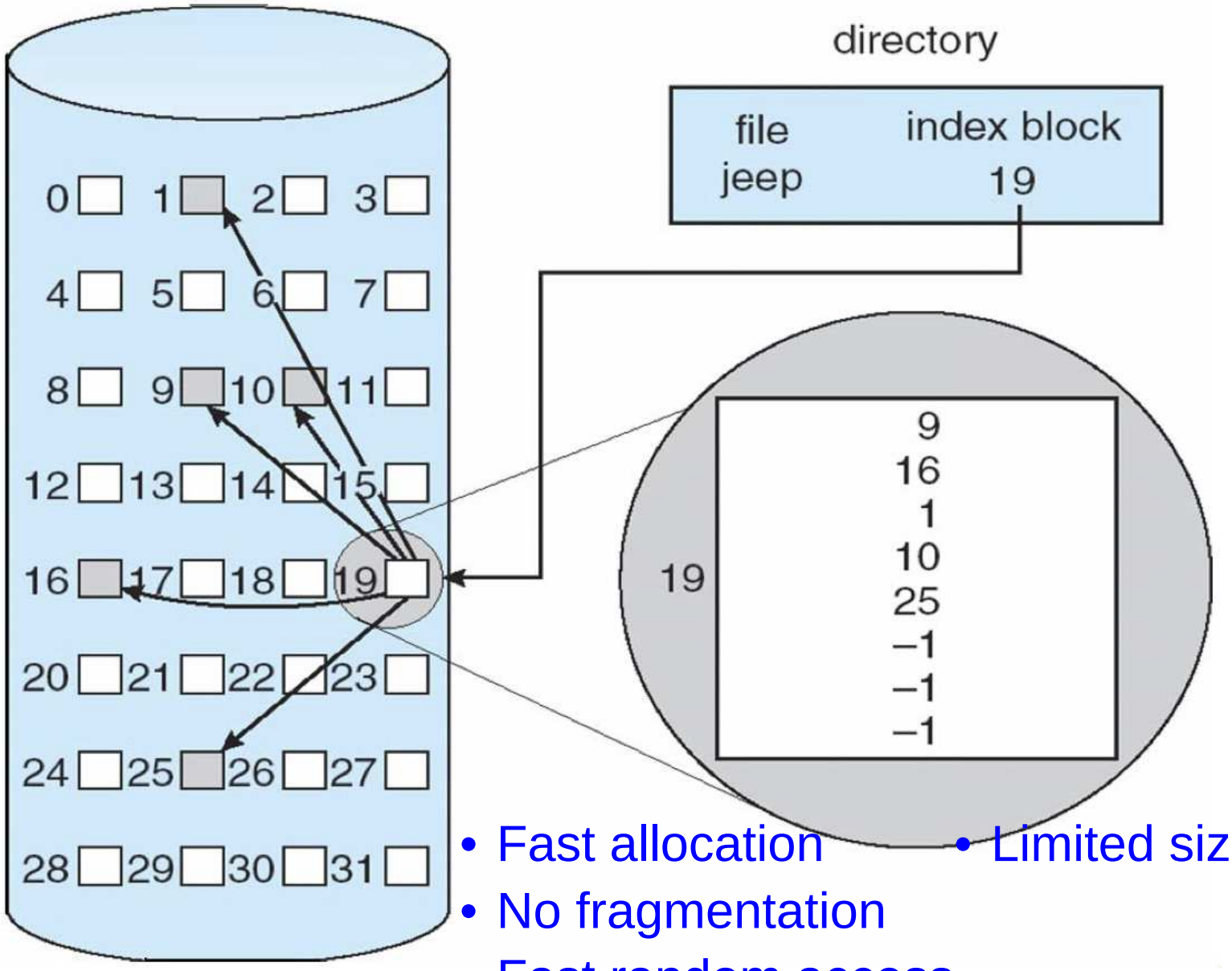
- Slow random access
- Easily corrupted

Linked Allocation with File-Allocation Table



avoids hopping around the disk to find the middle of a file

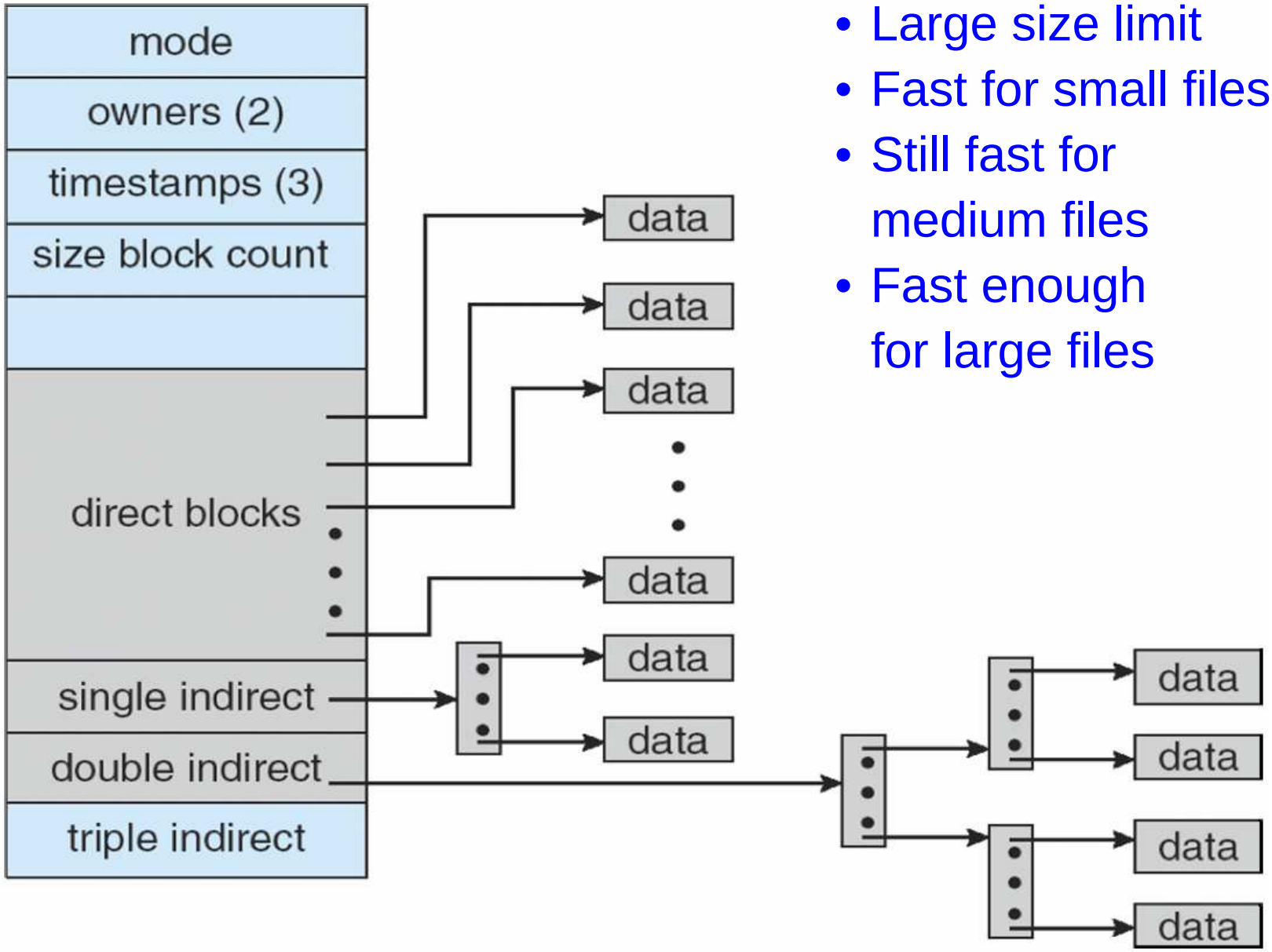
Indexed Allocation



- Fast allocation
- No fragmentation
- Fast random access

- Limited size

Multilevel Indexed Allocation



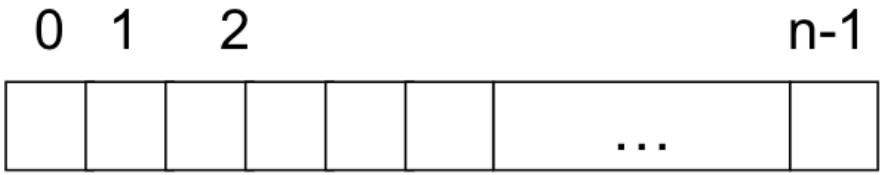
- Large size limit
- Fast for small files
- Still fast for medium files
- Fast enough for large files

Finding Empty Space

Some popular choices:

- Bitmap
- Linked list
- Indexed allocation

Bitmap



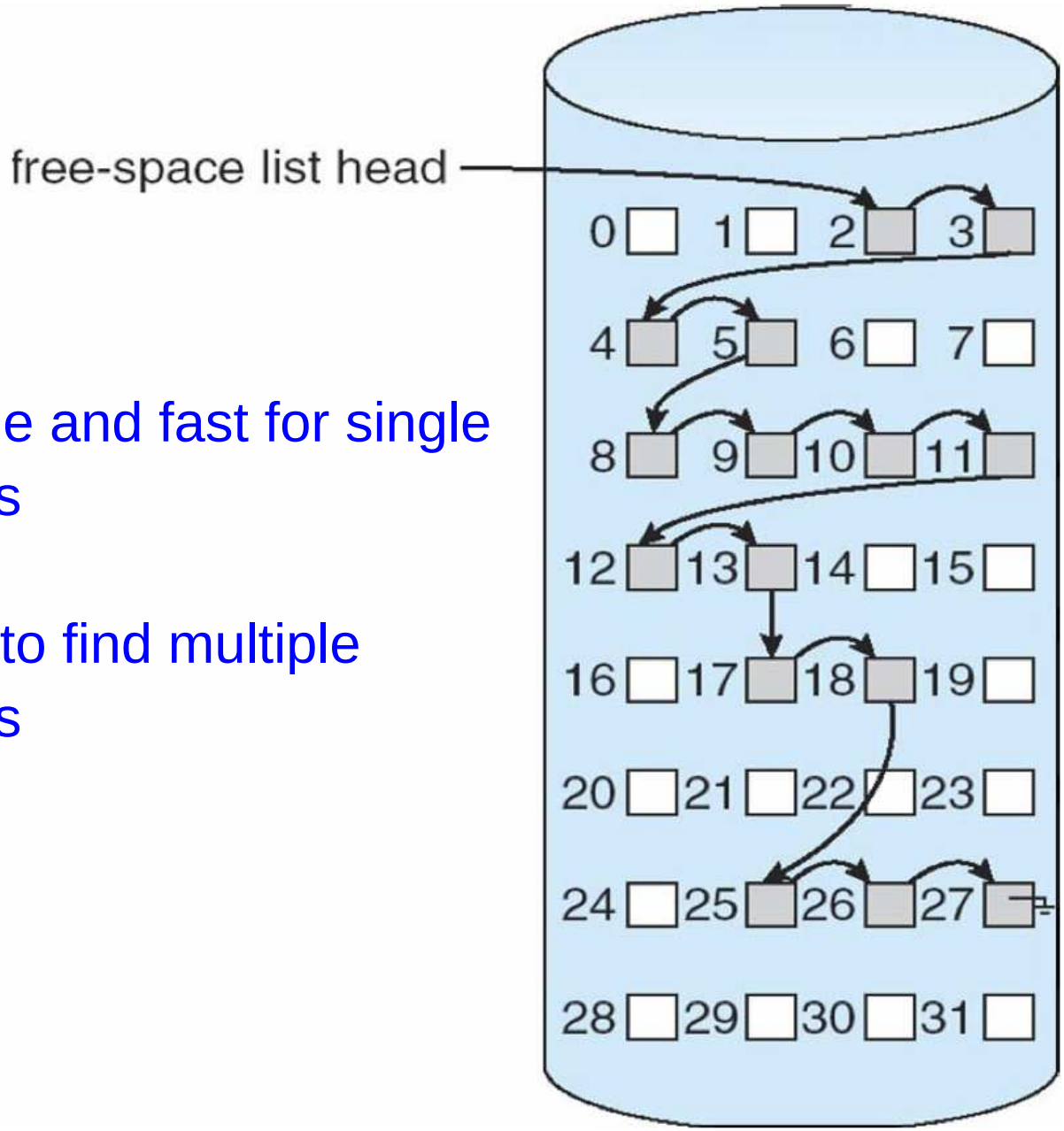
bit[*i*] = 0 ⇒ block[*i*] free

1 ⇒ block[*i*] occupied

- Simple and fast for small disks
- Large for large disks

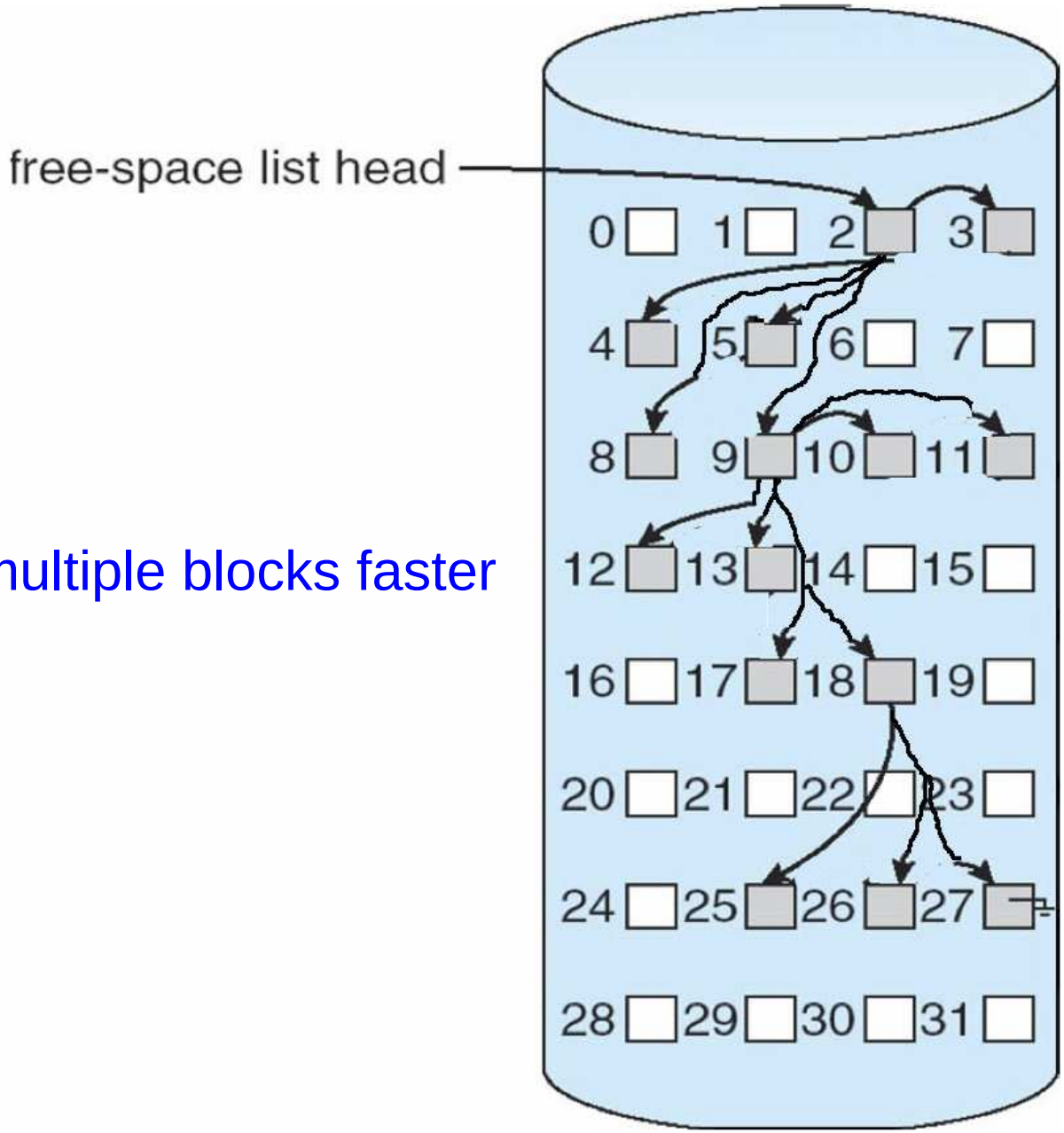
Linked List

- Simple and fast for single blocks
- Slow to find multiple blocks



Grouping

Finds multiple blocks faster



Recovery

What if the power goes out in the middle of a disk update?

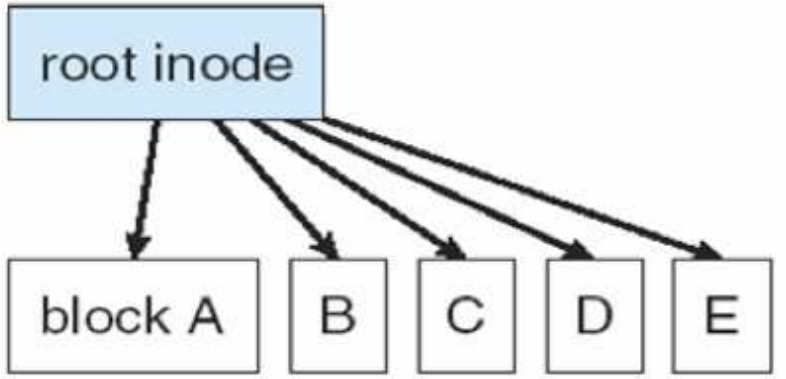
- Check disk and guess at repair: **`fsck`**
- Journaliing file systems: just retry

Journaling

For metadata tasks:

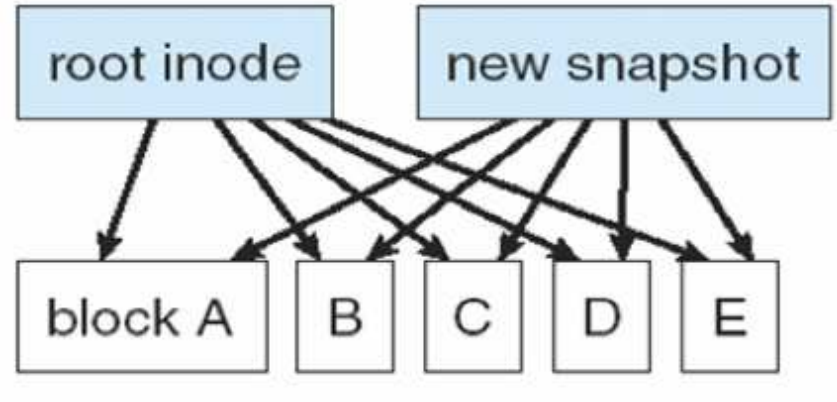
- Instead of modifying the disk, add to an on-disk queue to describe what action should be taken
- Each element stays in the queue until it is really done
  - If interrupted, can try again
- When mounting a journaling filesystem, if the queue is not empty, then perform all the leftover actions

Cheap Snapshots



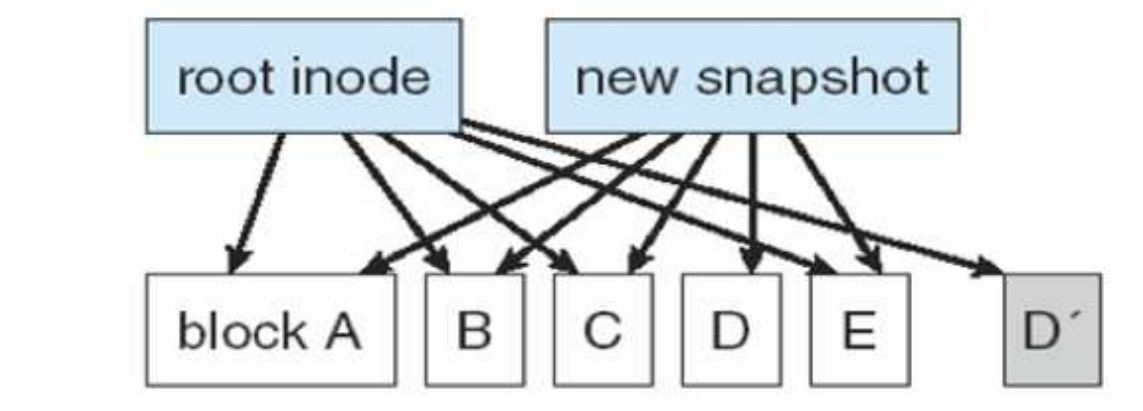
original content

Cheap Snapshots



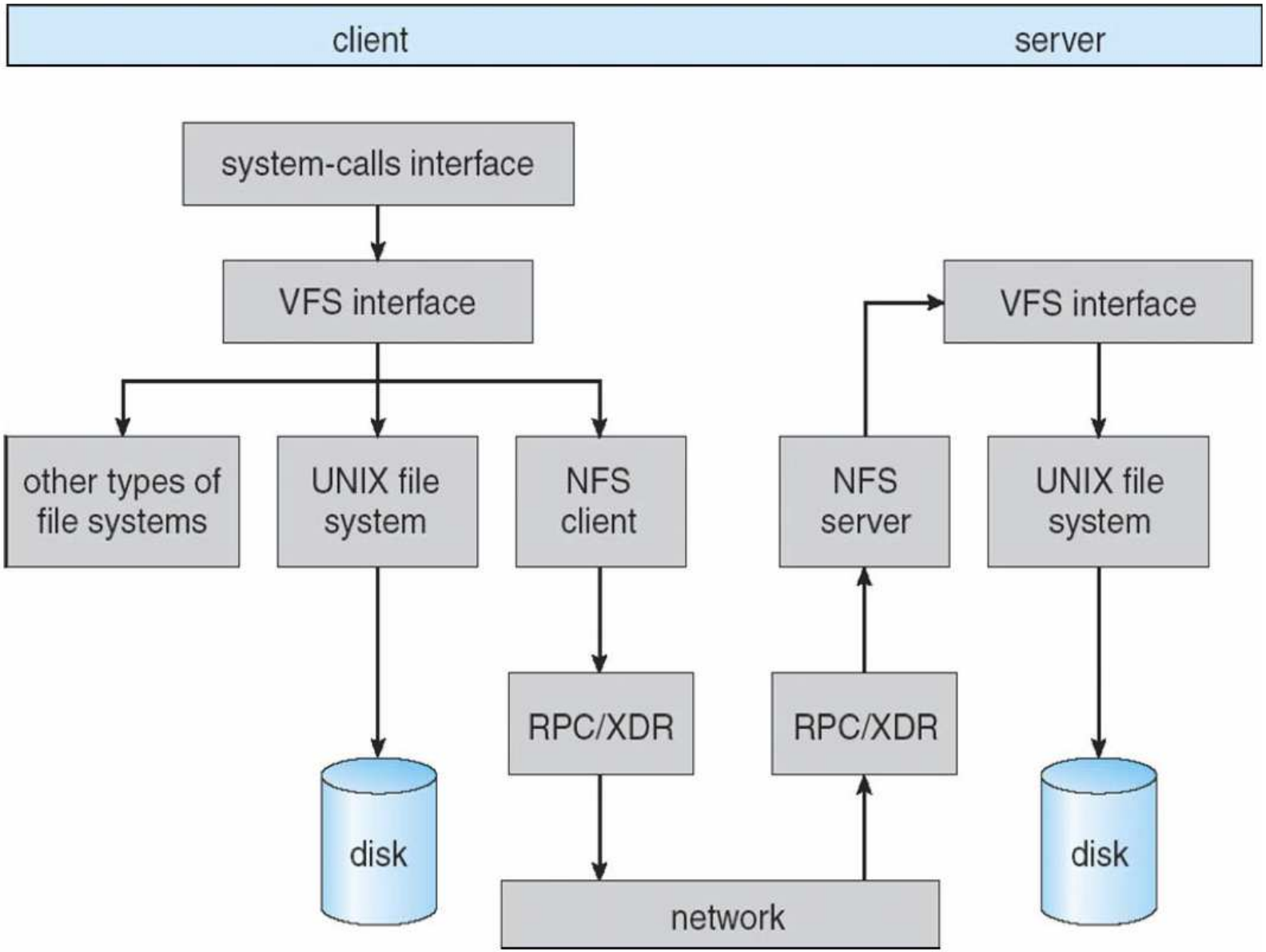
snapshot copies metadata

Cheap Snapshots



copy-on-write preserves snapshot

Network File System (NFS)



Network File System (NFS)

- NFS protocol very similar to Unix system calls
  - Except path resolution, which works on ***vnode***--name pairs
- Caching at clients improves performance
  - File content: client checks with server
  - Metadata: server sends change notifications
- Write preserved at client until reported done by server

Summary

- File systems must efficiently support many small files and a few large files
- Many file-system implementation ideas are similar to virtual memory
  - Contiguous allocation is simple; suffers from fragmentation and problems growing files
  - Indexed allocation is similar to page tables
  - Free space can be managed using a bitmap or a linked list
- *Major difference between disk and memory:* after reading a disk block, the next block can be read very cheaply