CSE 3320
Operating Systems
File Systems Management and Optimizations

Jia Rao

Department of Computer Science and Engineering

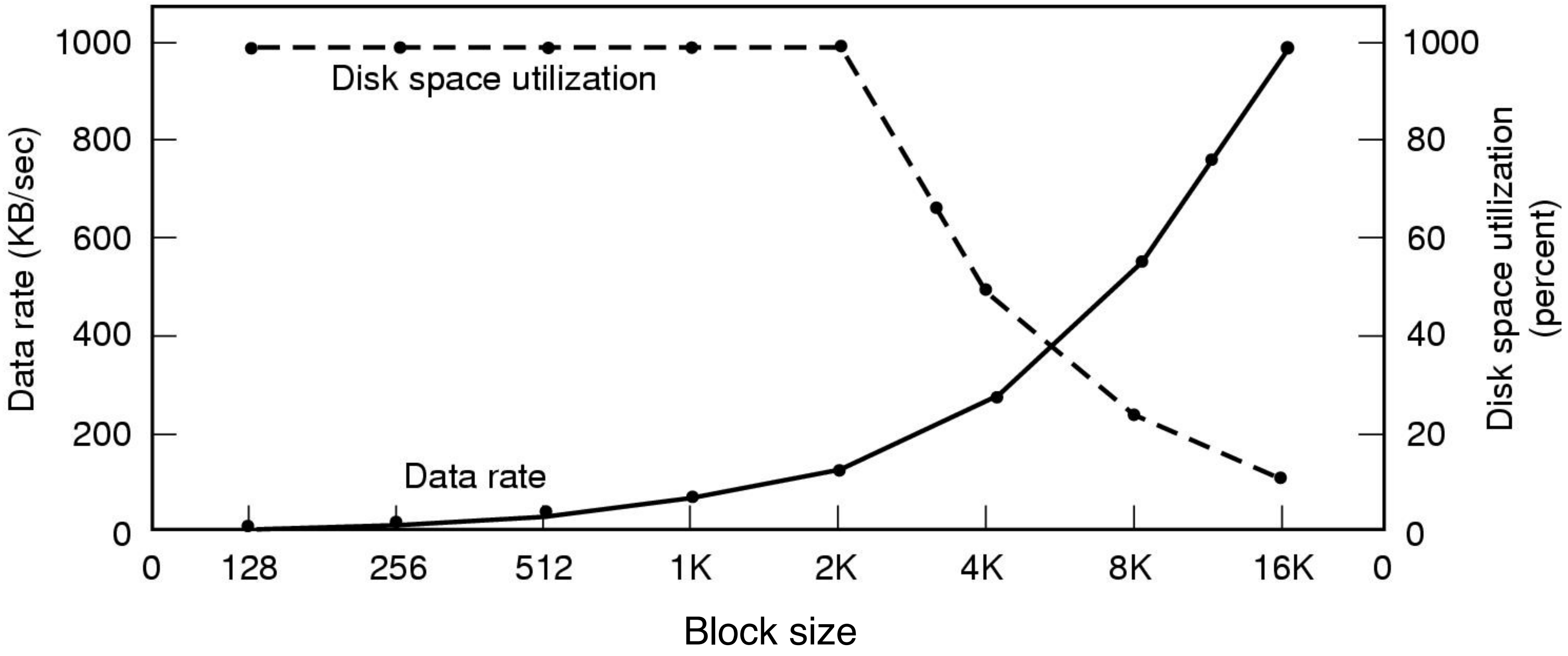
http://ranger.uta.edu/~jrao

Recap of the Previous Class

- Implementing files
  - Contiguous allocation
  - Linked allocation
  - FAT
  - Indexed allocation (I-node)
- Implementing directories
- Sharing files
- File system management and optimizations

Disk Space Management – Block Size

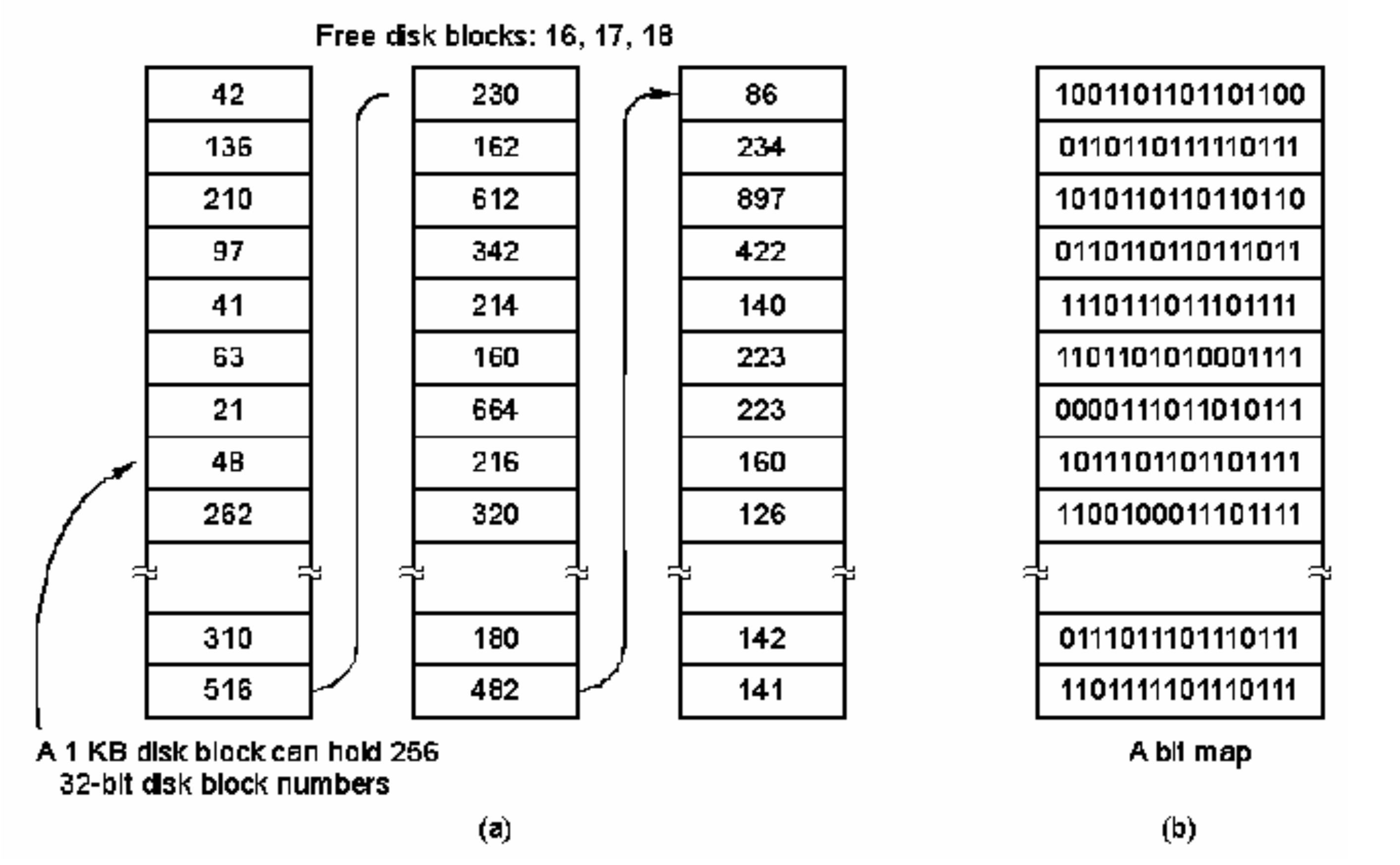
- All file systems chop files to fixed-size non-adjacent blocks
- Block size is a trade-off of *space utilization* and *data rate*



- Dark line (left hand scale) gives data rate of a disk
- Dotted line (right hand scale) gives disk space efficiency
- 60%-70% files below 4KB, 93% of the disk occupied by 10% largest files

Disk Space Management – Tracking Free Blocks

- How to keep track of free blocks?



(a) Storing the *free* list on a linked list. (b) A bit map

Example of Tracking Free Blocks

- Consider a 16-GB disk, 1-KB block size, 32-bit disk block number

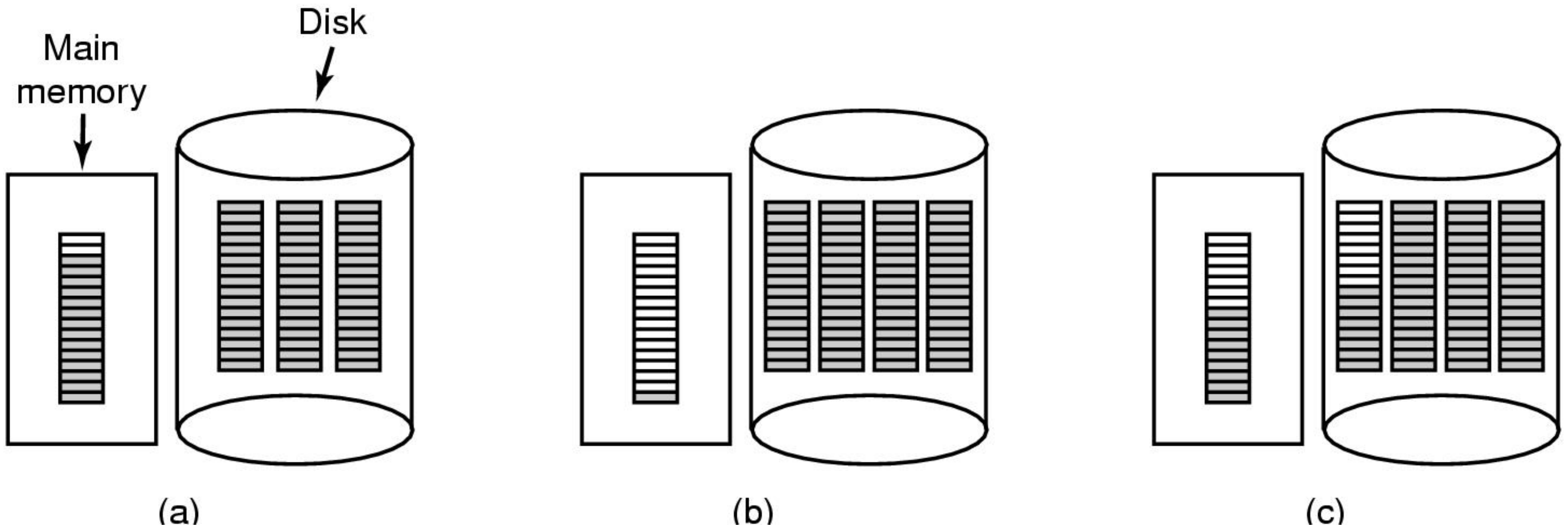
  if all blocks are empty, how many blocks in the free list and in the bit map, respectively? Which one uses less space? But what if the disk is nearly full?

  Linked list: (16 * 2^30 / 2^10) *4 = 64MB
  Bit map: (16 * 2^30/2^10)/8 = 2MB

How much information should be stored in the memory for each scheme?

Linked list: 1KB
Bit map: 2MB

An Issue and Optimization



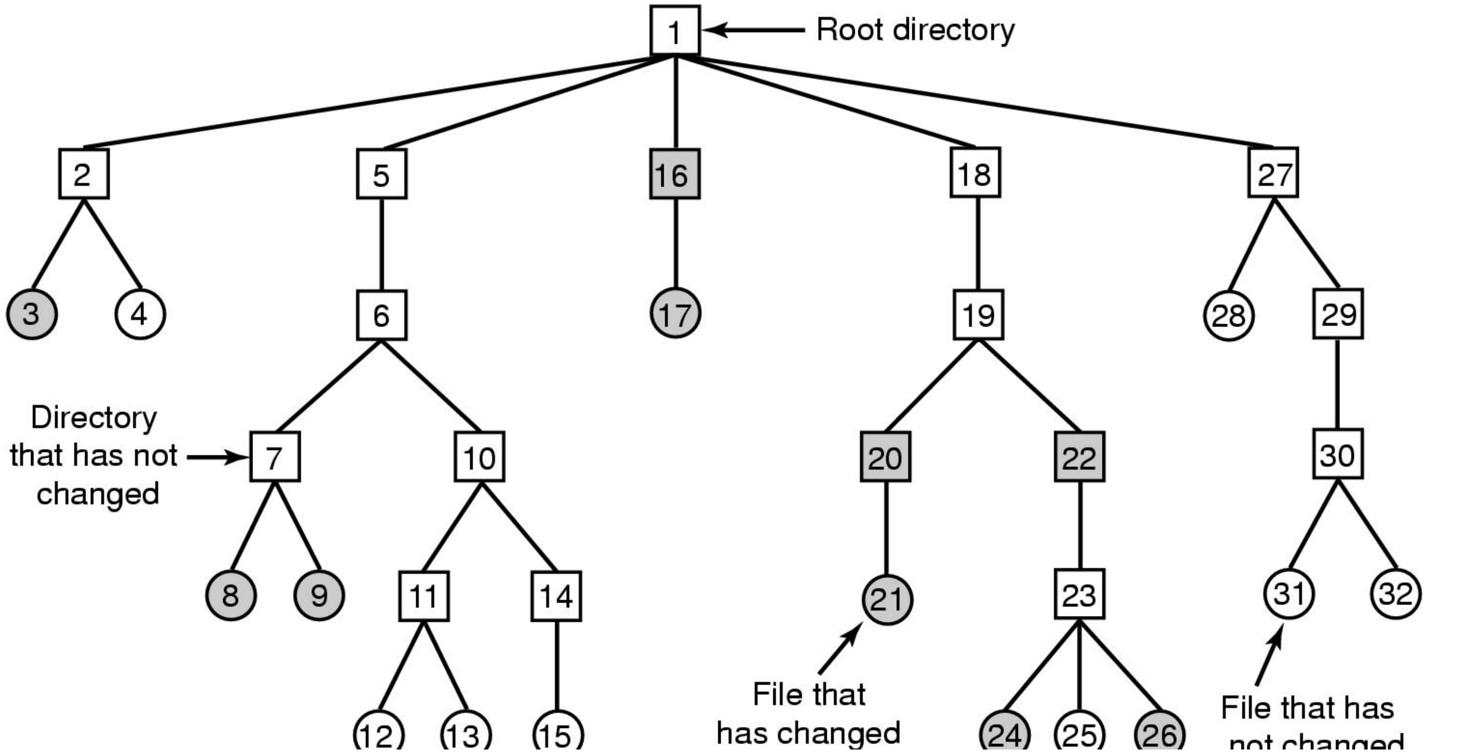
(a) An almost-full block of pointers to free disk blocks in memory and three blocks of pointers on disk. (b) Result of freeing a three-block file. (c) An alternative strategy for handling the three free blocks. The shaded entries represent pointers to free disk blocks.

File System Reliability (1)

- Physical dumping: starts at block 0, writes all the disk blocks onto the output tape in order

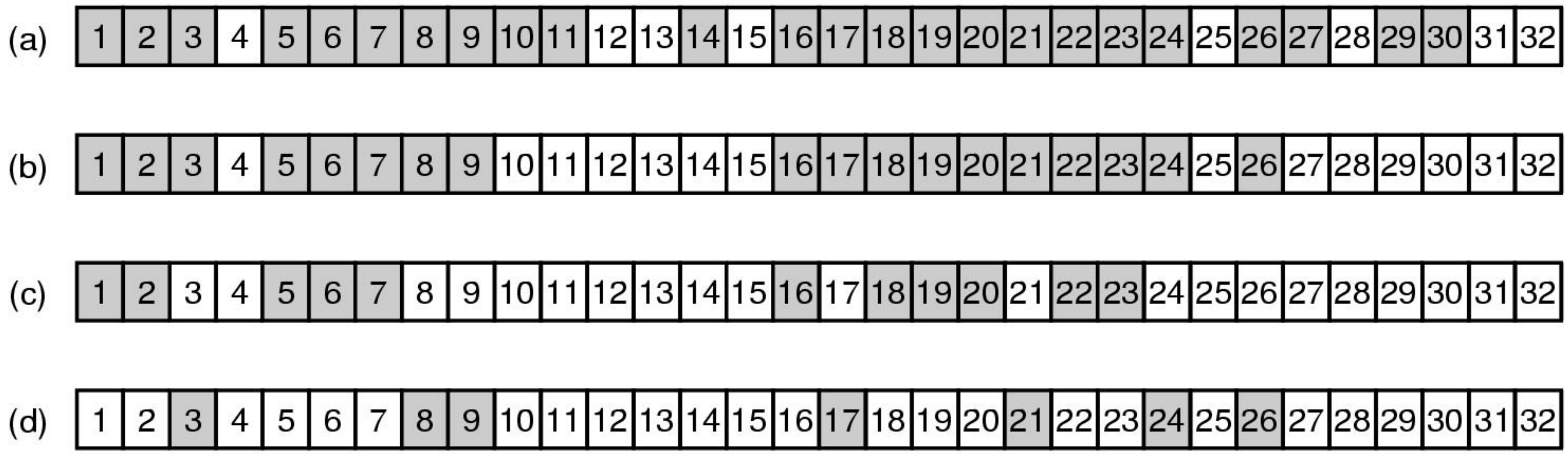
- Logical dumping: starts one or more specified directories and recursively dumps all files and directories found there that have changed since some given based date (e.g., the last backup from an incremental dump or system installation for a full dump)

File System Reliability (2)- logical dump



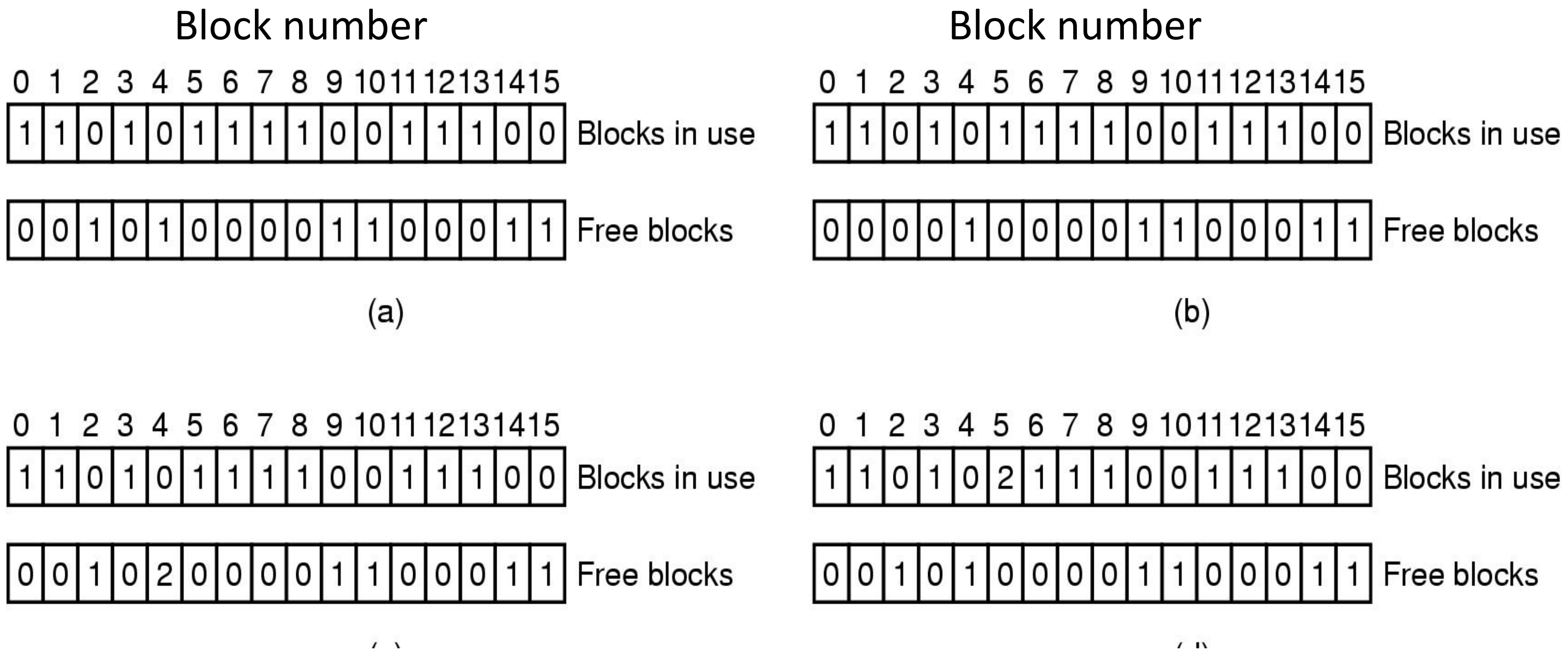
- Squares are directories
- Circles are files
- Shaded items have been modified

File System Reliability (3)- logical dump



- Phase 1: Mark all directories and modified files
- Phase 2: Unmark directories with no modified files
- Phase 3: Dump modified directories
- Phase 4: Dump modified files

File System Consistency (1)- Block Check



Block number

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Blocks in use	1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
Free blocks	0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1

(a)

Block number

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Blocks in use	1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
Free blocks	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1	1

(b)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Blocks in use	1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
Free blocks	0	0	1	0	2	0	0	0	0	1	1	0	0	0	1	1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Blocks in use	1	1	0	1	0	2	1	1	1	0	0	1	1	1	0	0
Free blocks	0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1

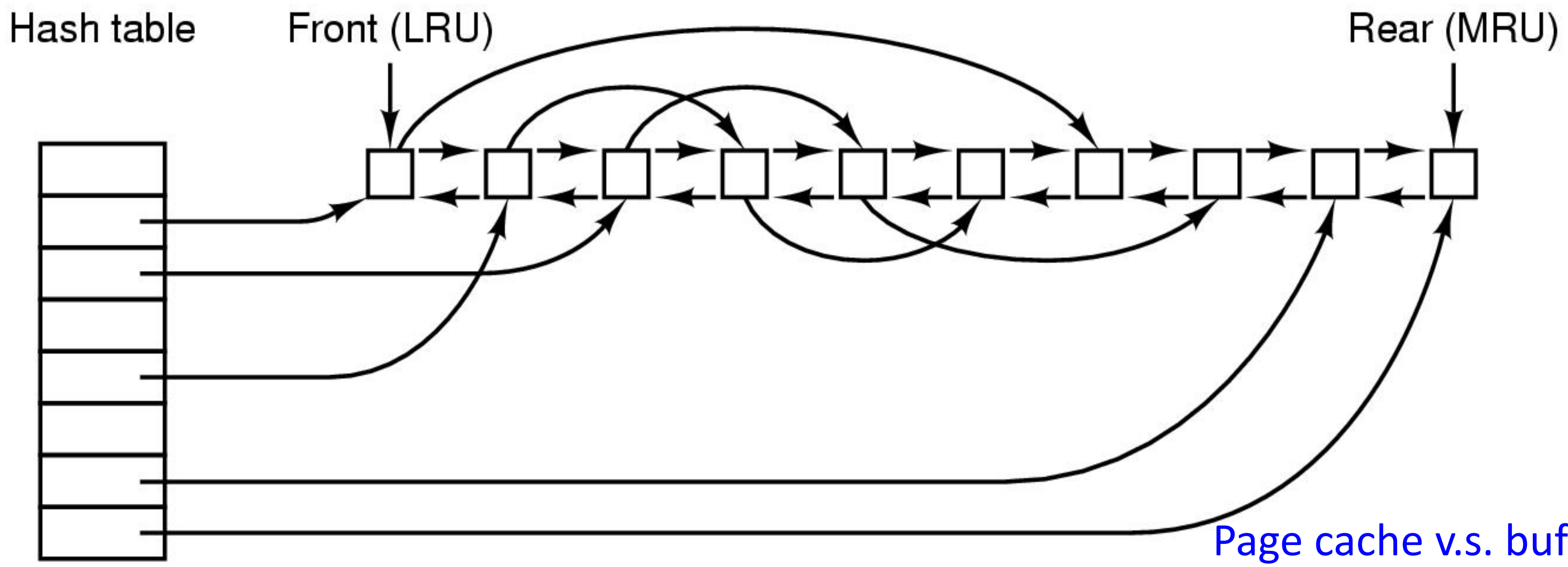
- File system states
  - (a) consistent
  - (b) missing block
  - (c) duplicate block in free list
  - (d) duplicate data block

File System Consistency (2)- File Check

- Use a table of counters each counts the occurrence of files
- Only count the hard links
- Compare the occurrence of files with the info stored in its i-node

File System Performance - Caching

- Cache: a collection of blocks that logically belong on the disk but are being kept in memory for performance reasons
  - Hash the device and disk address and look up the result in a hash table with *collision chains*
  - Cache references are relatively infrequent



Page cache v.s. buffer cache

The block cache data structures with a *bi-directional usage list*

Why LRU is undesirable when consistency is an issue if the system crashes?

File System Performance – Caching II

- Cache & Consistency
  - UNIX system call sync() every 30s
  - UNIX system checks dirty_background_ratio and dirty_ratio
  - MS-DOS strategy write-through

File System Performance – Block Read Ahead

- Block Read Ahead works well for files that are being read sequentially
  - Spatial locality

    `LINUX_SRC/mm/readahead.c`

    - How to detect a sequential access pattern?
    - How much data to be prefetched?
    - At what time the prefetching requests should be issued?

Summary

- Block size
- Free block management
- File system reliability
- File system consistency
- File system performance
  - Caching
  - Disk block readahead