The Buffer Cache

Objectives

- Discuss the Unix kernel architecture
- To learn the use of buffer cache
- Learn the buffer pool structure
- To learn how reading and writing disk blocks is performed with buffer cache

The Unix Kernel Architecture

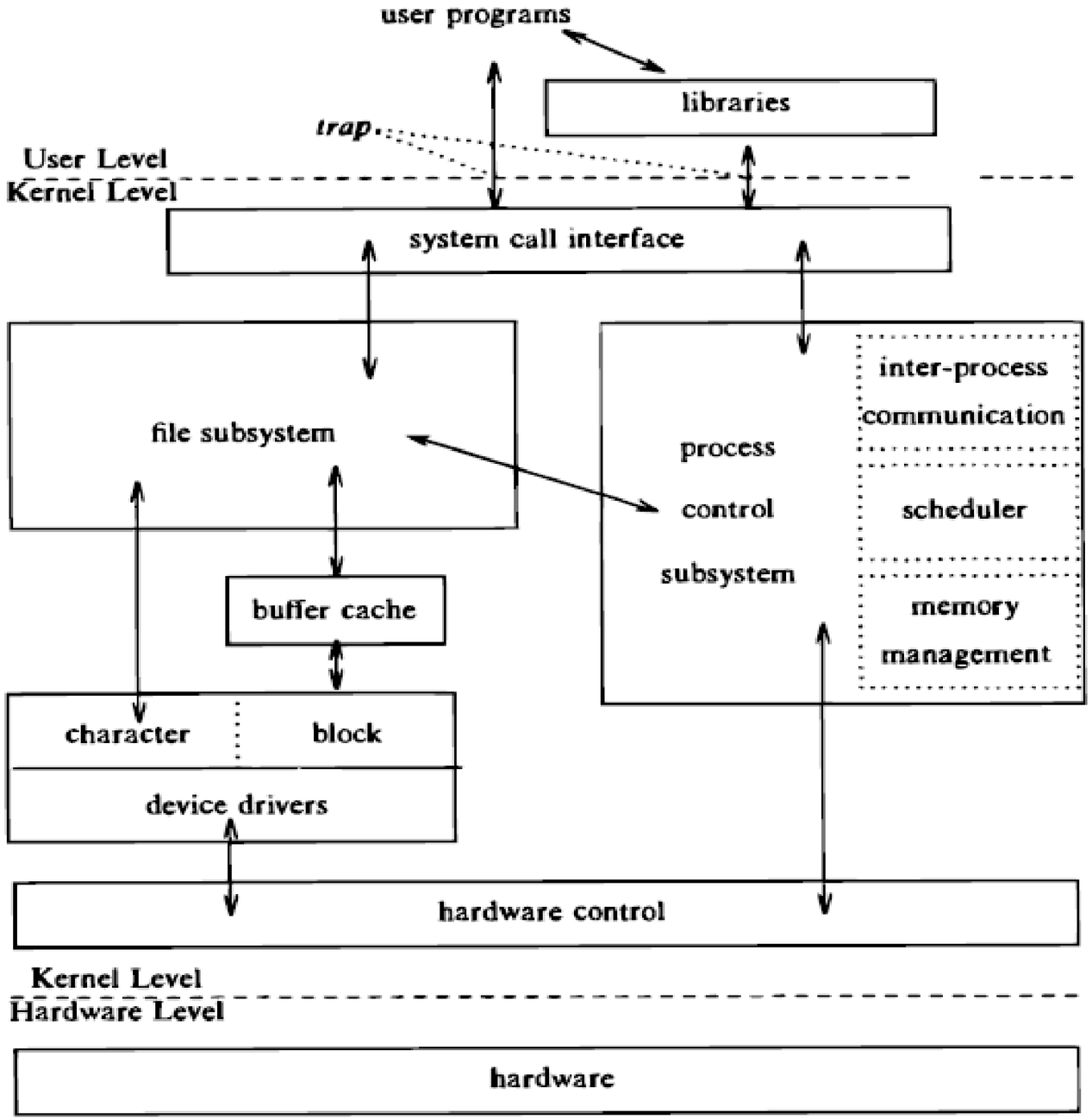


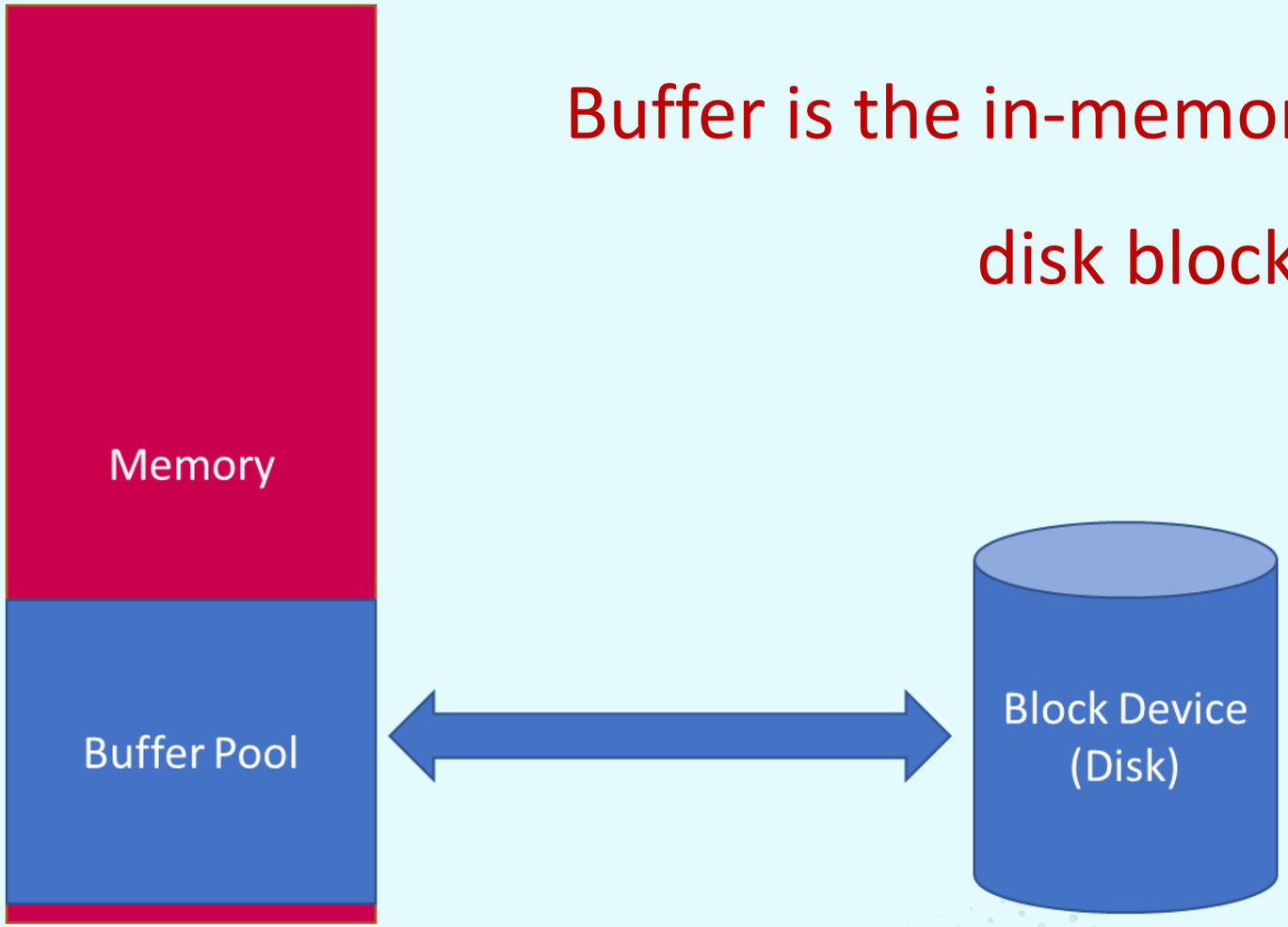
Figure 2.1. Block Diagram of the System Kernel

The Buffer Cache

- The kernel could read and write directly to and from the disk for all the file system accesses.
- Poor response time and throughput because of the slow disk transfer rate.

The kernel therefore attempts to minimize the frequency of disk access by keeping a pool of data buffers, called the buffer cache.

Buffer Cache



Buffer is the in-memory copy of the disk block.

Buffer Header

- During system initialization, the kernel allocates space for a number of buffers.
- Configurable based on memory size and performance.

Buffer has two parts

A memory array – Contains data from the disk

Buffer header – Identifies the buffer

Buffer Header

- Disk is a block device
  - Reads and writes as block of data
  - Buffer can store one block of data read from the disk.
  - Data in the buffer corresponds to data in a logical block.

Kernel identifies the buffer contents by examining the identification field in the buffer header.

A disk block can never map to more than one buffer at a time.

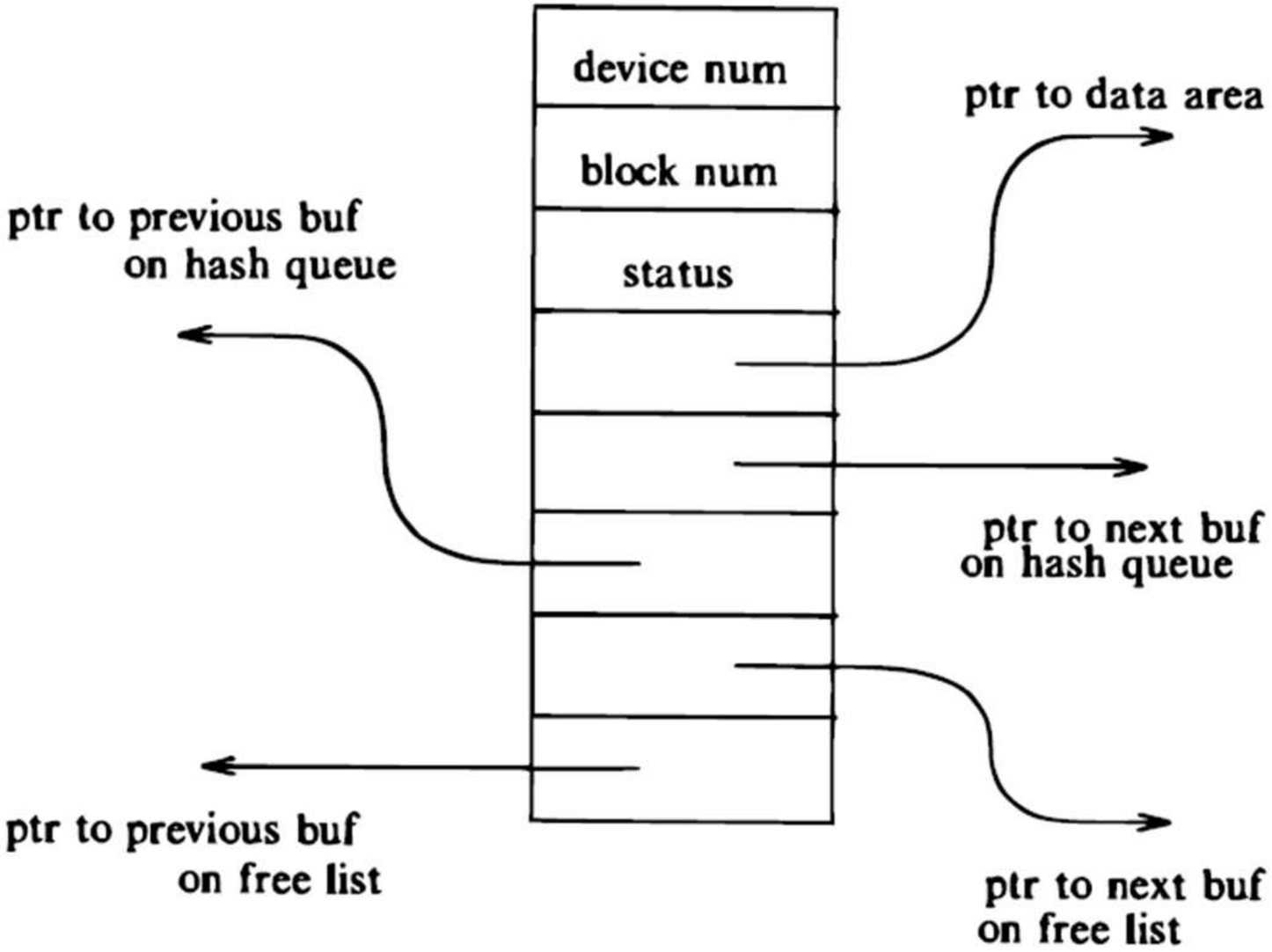


Figure 3.1. Buffer Header

The Buffer Header

- Device Number : Logical file system number
- Block Number: Block number of the data on the disk
  - Status
    - Whether the buffer currently locked or busy
    - Whether contains valid data
    - Delayed write or not
    - The kernel is currently reading or writing
    - Is there a process is waiting for the buffer to become free.

Structure of Buffer Pool

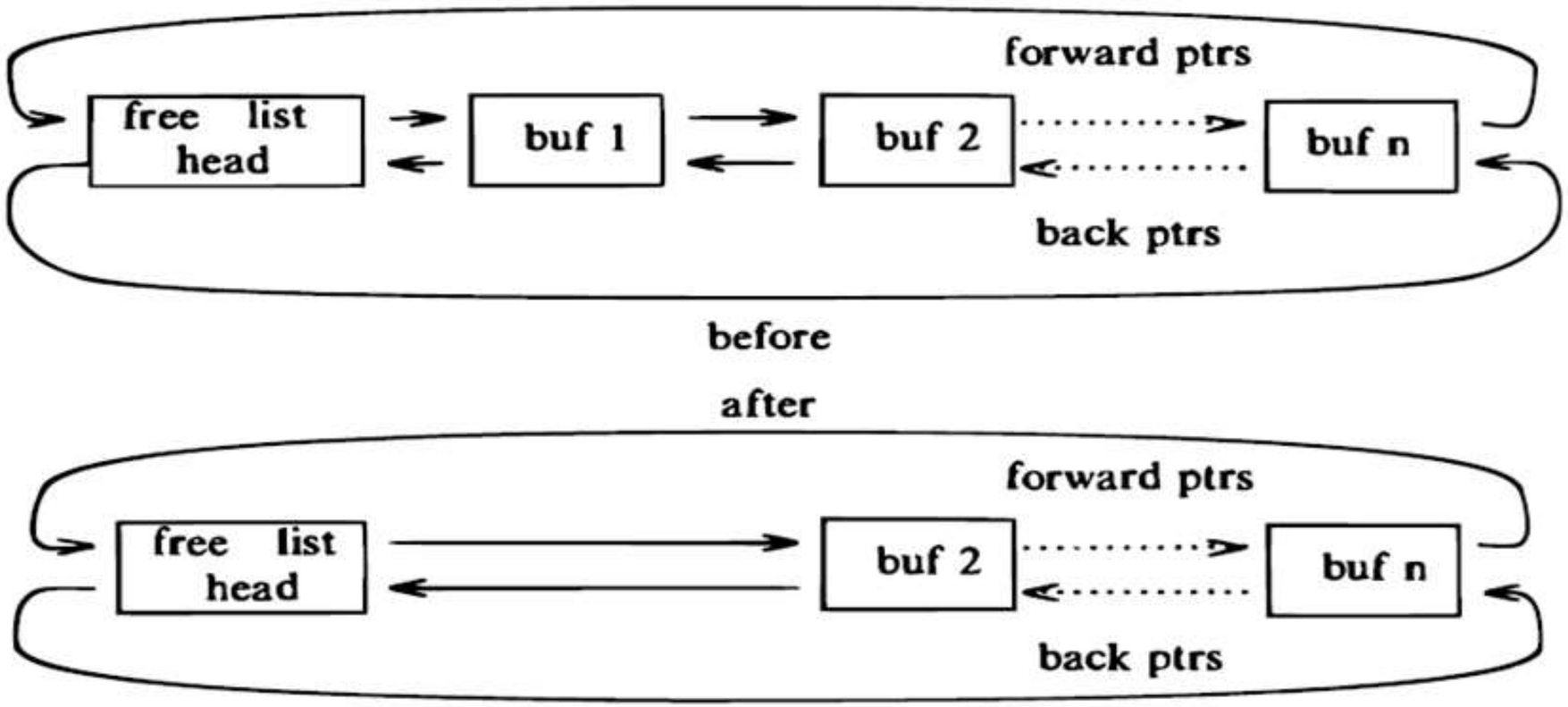


Figure 3.2. Free List of Buffers

The Buffer Pool

- The kernel follows the least recently used (LRU) algorithm for the buffer pool.
- The kernel maintains a free list of buffers that preserves the least recently used order.
- Implemented as a doubly circular link list of buffers with a dummy buffer header, marks the beginning and end.

The Buffer Pool

- All the buffers are put on the free list when the system is booted.
- When the kernel wants any buffer, it takes it from the head of the free list.
- The used buffers, when become free, are attached to the end of the list, hence the buffers closer and closer to the head of the list are the most recently used ones.

Reading/Writing into Buffers

- When the kernel access a disk block, it searches the buffer with appropriate device number.
- Instead of searching the entire pool, Unix organizes the buffers into separate queues, hashed as a function of device and block number.
- Then the hash is arranged as a circular, doubly , link list.
- The hash function used must be simple so that the performance does not suffer.

Buffers on the Hash Queue

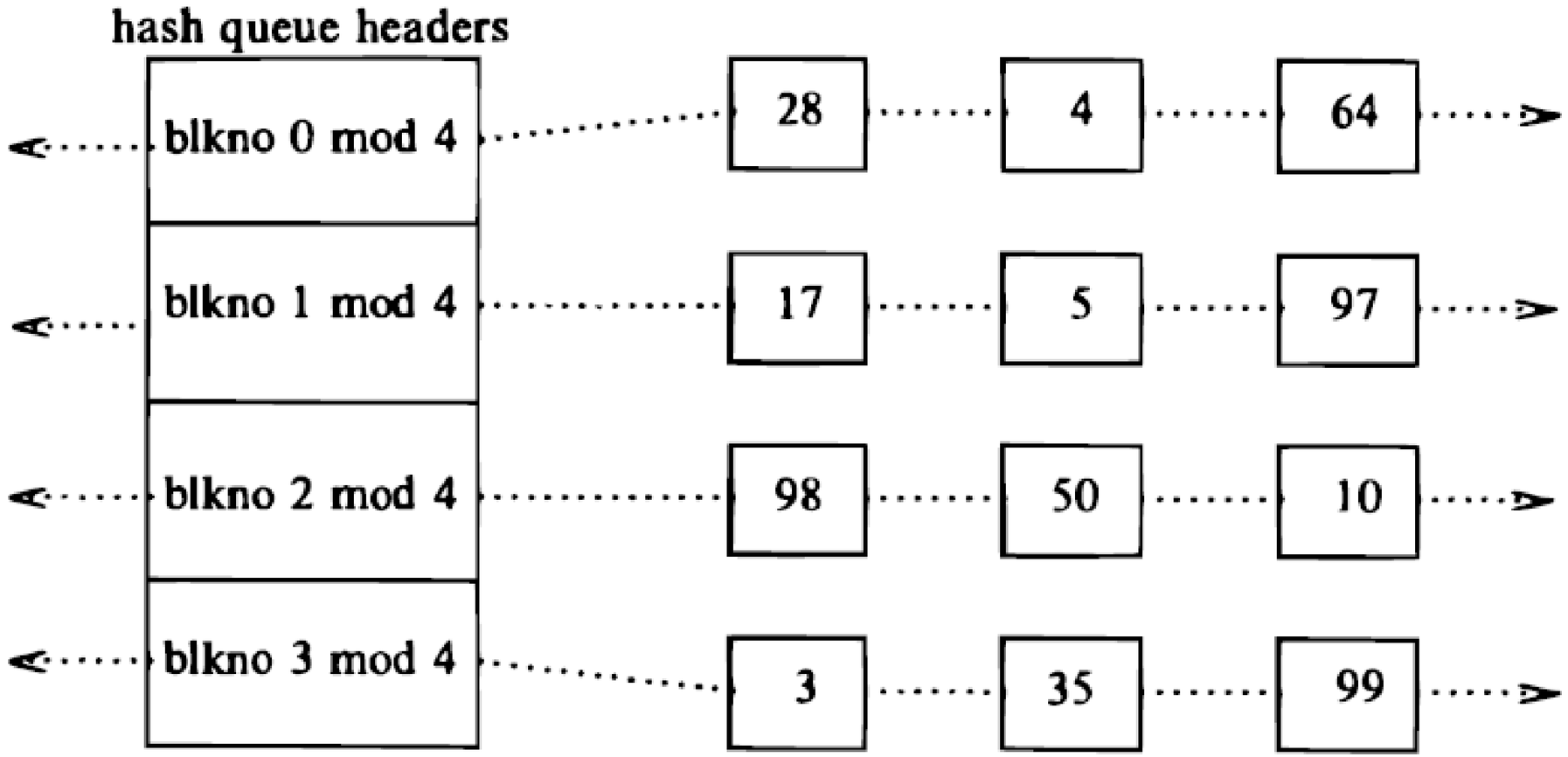


Figure 3.3. Buffers on the Hash Queues

Reading Disk Block

- If a process wants to read data from file
  - ✓ Kernel determines the file system number and block number
  - ✓ Then it checks whether the block is in the buffer pool
  - ✓ If not, assigns a free buffer
- When to write data to file
  - ✓ Kernel checks whether the block is in the buffer pool
  - ✓ If not, assign a free buffer for that block.

Reading/Writing Disk Block

- While reading or writing getblk() searches the buffer cache.
- If it is there in the cache, kernel can return it immediately without physically accessing the disk block.
- If not in cache, kernel calls disk driver and initiates the I/O operation.
- Contents fetched are then placed into the allocated buffer.

Reading/Writing Disk Block

- Writing to disk can be
  - ✓ Asynchronous Write : Kernel starts disk operation immediately but does not wait for completion
  - ✓ Delayed Write : Kernel puts off, the physical write to disk as long as possible.

Advantages

- ✓ Uniform disk access => system design simpler
- ✓ Use of the buffer cache can reduce the amount of disk traffic.
- ✓ Single image of disk blocks contained in the cache => helps insure file system integrity.

Disadvantages

- ✓ Delayed Write
  - ✓ Vulnerable to crashes that leaves system in incorrect state
  - ✓ Maintenance of extra copy

Summary

- Learned the use of buffer cache in improving the performance of system.
- The buffer cache structure and buffer header.
- The buffer pool arrangement and implementation.
- Reading and writing disk blocks.
- Advantages and disadvantages of buffer cache.

References

“The design of Unix Operating System”, Maurice J Bach

http://unixbyrahul.50webs.com/unix3.html