CS343: Operating System

Memory Management

Lect26 : 06th Oct 2023

Dr. A. Sahu

Dept of Comp. Sc. & Engg.

Indian Institute of Technology Guwahati

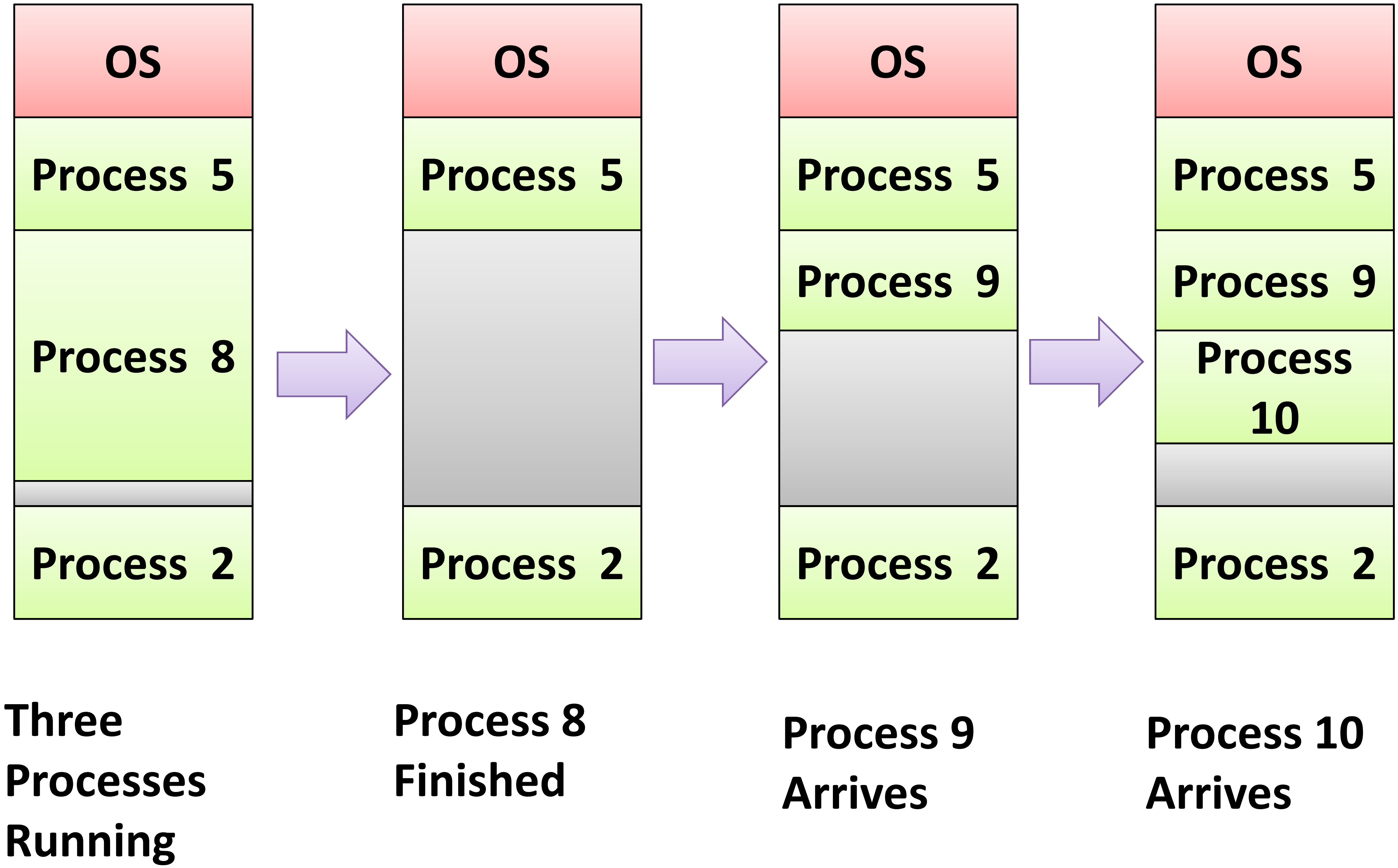
Outline

- Motivations for Memory management
- Memory Management
- High level (Top Down Approach)
  - Process Continuous Allocation
  - Segment continuous Allocation
  - Paging
  - Demand paging
  - Frame Allocation

Memory Allocation

- When a process arrives
  - OS allocate memory from a hole large enough to accommodate it
- Process exiting frees its partition, adjacent free partitions combined
- OS maintains information about
  - a) allocated partitions
  - b) free partitions (hole)

Process Continuous Memory Allocation



Dynamic Storage-Allocation Problem

- How to satisfy a request of size ***n*** from a list of free holes?
  - **First-fit**: Allocate the ***first*** hole that is big enough
  - **Best-fit**: Allocate the ***smallest*** hole that is big enough; must search entire list, **unless ordered by size**
    - Produces the smallest leftover hole
  - **Worst-fit**: Allocate the ***largest*** hole; must also search entire list
    - Produces the largest leftover hole
- First-fit and best-fit better than worst-fit in terms of speed and storage utilization

Fragmentation

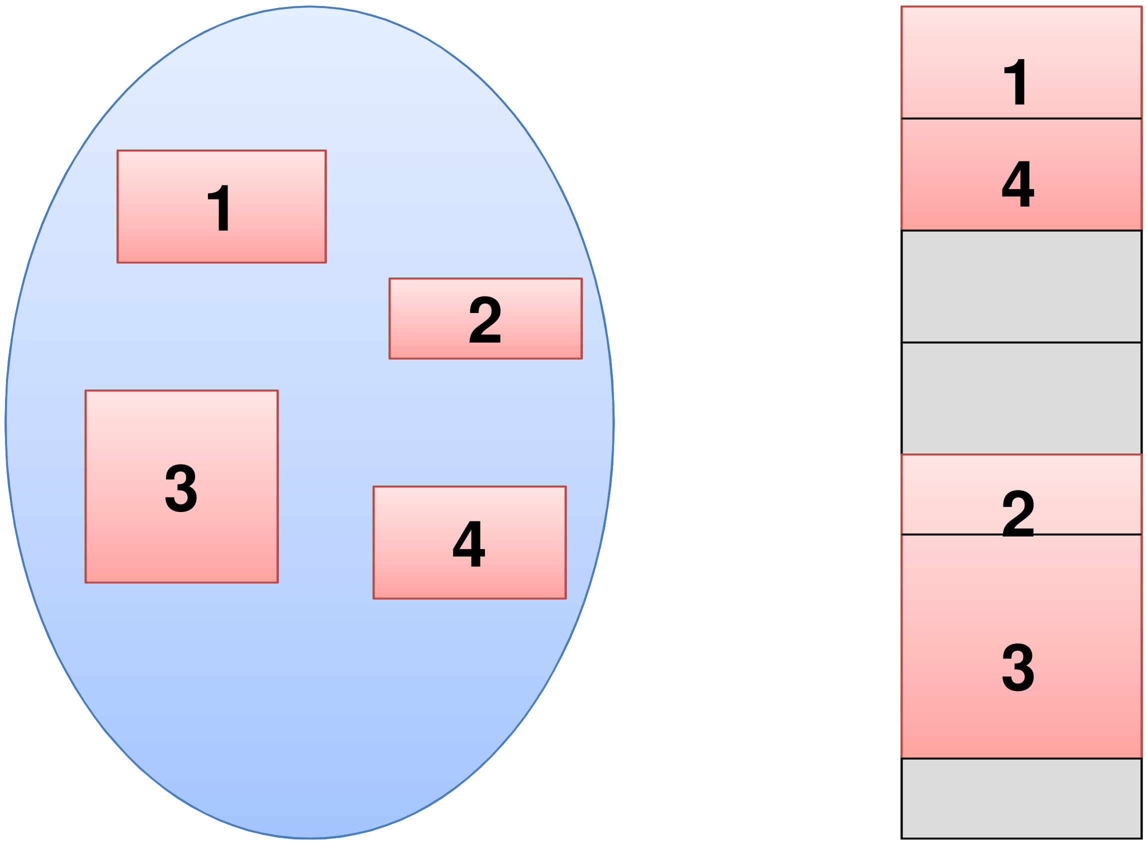
- **External Fragmentation** – Total memory space exists to satisfy a request, but it is not contiguous
- **Internal Fragmentation** – allocated memory may be slightly larger than requested memory
  - This size difference is memory internal to a partition, but not being used

Fragmentation (Cont.)

- Reduce external fragmentation by **compaction**
  - Shuffle memory contents to place all free memory together in one large block
  - Compaction is possible *only* if relocation is dynamic, and is done at execution time
  - I/O problem
    - Latch job in memory while it is involved in I/O
    - Do I/O only into OS buffers
- Now consider that backing store has same fragmentation problems

Segmentation

Logical View of Segmentation



user space **physical memory space**

Segmentation and Paging

- Divide the program into smaller block call segments : User view, already segmented
  - Finer granularity, less fragmentation
    - **Mustard in Bag Vs Brinjal in Bag**
- Divide the program in to smaller but uniform size unit called page
  - A program may contain many pages
  - Last page of program may be partially filled
- Divide the memory in to smaller size units called Frame
- Page get mapped to Frame

Paging

Paging

- Physical address space of a process can be noncontiguous; process is allocated physical memory whenever the latter is available
  - Avoids external fragmentation
  - Avoids problem of varying sized memory chunks
- Divide physical memory into fixed-sized blocks called **frames**
  - Size is power of 2, between 512 bytes and 16 Mbytes , **getpagesize() in C → Demo**

Paging

- Divide logical/program memory into blocks of same size called **pages**
- Keep track of all free frames
- To run a program of size ***N*** pages, need to find ***N*** free frames and load program
- Set up a **page table** to translate logical to physical addresses
- Backing store likewise split into pages
- Still have Internal fragmentation

Memory Allocation: Top Down

Assume 4GB of RAM,
Avg Process Size 200MB
4 Segments/Process

OS
P5
P9
P8

P2

Finding Hole
First Fit
Worst fit
Best Fit

Very low complexity

Very High Mem Wastage

Process Continuous Allocation
10 process and 20 holes

Segment continuous Allocation
40 segment and 80 holes

low complexity

High Memory Wastage

Paging: 200MB process, 4KB page
Two Issue: pages and frames

Very Low Mem Wastage

High Complexity: 51200 pages/Process

Loading 51K pages in 10^6 frames

High Complexity: $4GB/4KB=2^{20}=10^6$ frames

SOL

Demand Paging Load the required 50-100 pages and load as an when necessary

4GB paged

SOL

Allocate 100-200 frames per process: reduce search Space

Thanks