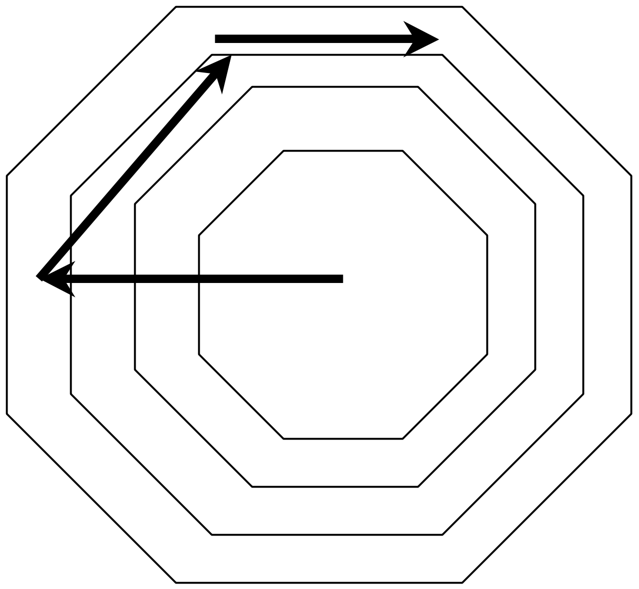
Outline

- Chapter 9: Memory management
- Chapter 10: Virtual Memory
- NUMA page placement by LaRowe
- HWP1 due, HWP2 assigned, HWA1 assigned
- Project milestone 1: Each student will submit their name, email address, and two or three areas of interest for a term project by email. This list will be distributed to the entire class. The goal is to match students with similar interests and form project teams. Students who have already formed teams make a single submission with all names and their area(s) of interest.

HWP 2: Disk scheduling

- Seek time: upto 2 sec
- Rotational latency: upto 2 sec
- Transfer rate: upto 2 sec (depends on the track)
- Repeat HWP1 and measure new performance numbers



Virtual Memory

- Address binding:
  Compile time - absolute code generated (MS Dos .com file)
- Load time (relocatable)
- Execution time: H/w support needed
- Logical and Physical address
  – MMU for run time mapping of virtual to physical address
- Dynamic linking and shared libraries
  – overlays

Paging

- Allows process physical page to be non-contiguous
  - Avoid fragmentation
  - Physical memory is broken into frames (h/w)
  - Logical memory is broken into pages (h/w)
  - Every address consists of page number and page offset. Page number is used as an index into page table

  - Page table base register (PTBR)
    - Increased memory access overhead

  - TLB (translation look-aside buffer)
    - Fast cache
    - Address-space identifiers (process protection)

Protection

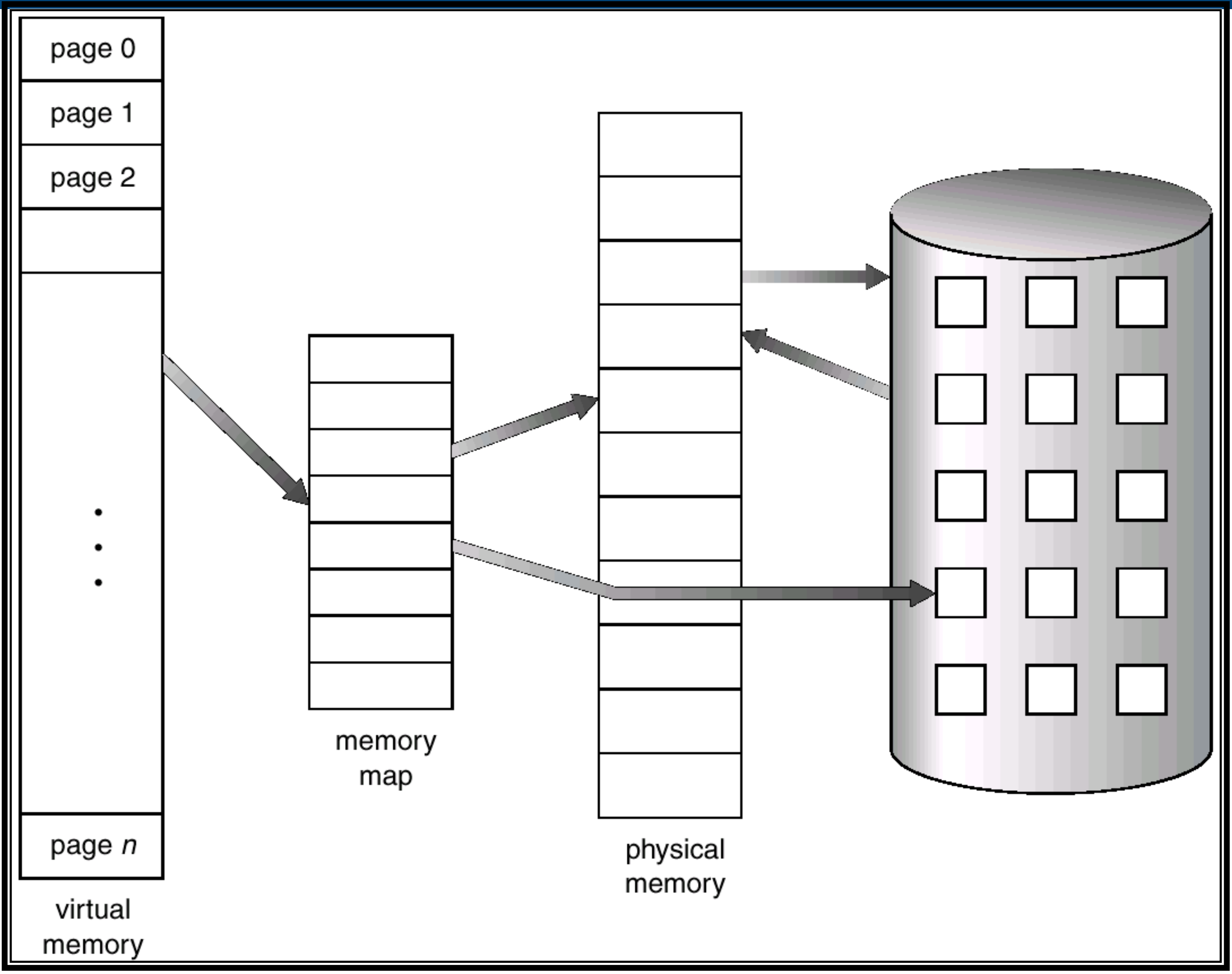
- Invalid bit
  - OS sets up this bit to generate trap

- Paging:
  - Hierarchical: unwieldy for large address spaces
  - Hashed page table
  - Inverted page table:
    - Store process:page-number
    - Table is sorted by physical address and lookups are done using virtual address
      - Hashing to reduce search space

Virtual memory

- separation of user logical memory from physical memory
  - Only part of the program needs to be in memory for execution
  - Logical address space can therefore be much larger than physical address space
  - Allows address spaces to be shared by several processes
  - Allows for more efficient process creation
- Virtual memory can be implemented via:
  - Demand paging
  - Demand segmentation

Virtual Memory



Demand paging

- Lazy swapper
  - Invalid pages generate page-fault trap
  - OS checks to see if it is indeed illegal
  - If valid, page it in
  - Once disk IO completely, schedule the process and resume execution
  - Performance implications

Process creation

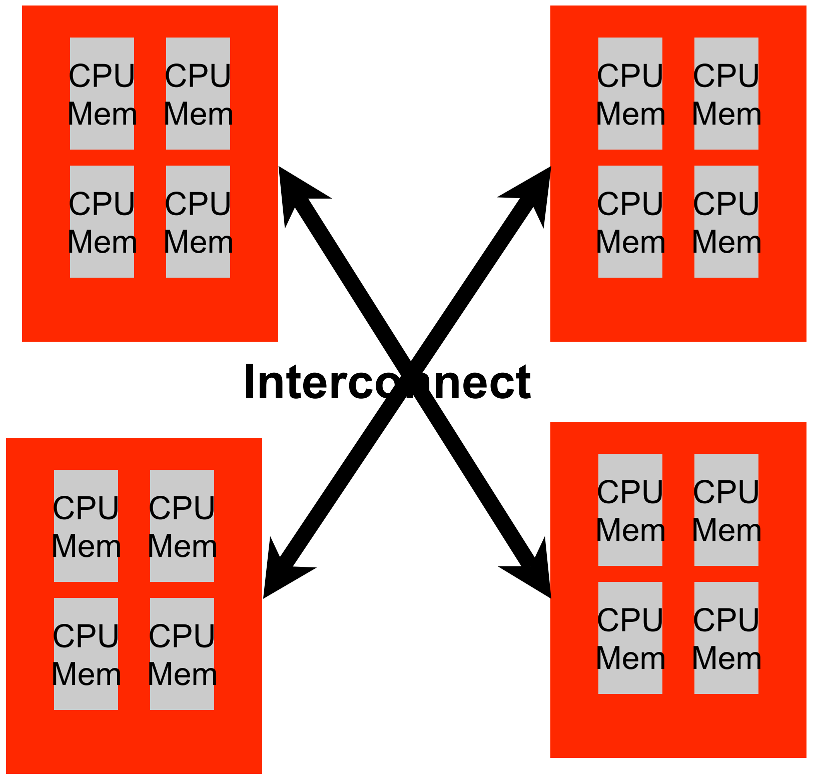
- Copy-on-write
  - Fork model duplicates process. OS shares the same copy and marks the pages invalid for write. Duplicate when any process tries to modify a page (trapper to the OS)
- Memory mapped files:
  - Virtual address space is logically associated with a file
- Page replacement:
  - Memory reference string
  - FIFO page replacement
    - Belady’s anomaly

Optimal page replacement

- Replace the page that will not be used for the longest period of time
- Approximations
  - Least recently used (LRU)
    - Counters: Update every page access
    - Stack of page numbers used
    - Additional reference bit algorithm
    - Second chance algorithm
    - Enhanced second-chance algorithm:
      - Second chance+reference bit
- Least Frequently used
- Most Frequently used
- Global vs local replacement
- Thrashing

NUMA

- Popular way to build scalable shared memory multi-processor
- Accesses to memory is not equal - page placement is important



OS vs application page placement

- Applications know best
- OS techniques can be leveraged by many applications

- How good are the tuneable page placement policies for a NUMA machine
- Too few migrations: pay remote access cost
- Too many migrations: pay page movement overhead
- Tradeoff local access for page movement
- Show that the policies are robust - a general setting applicable to a number of different scenarios