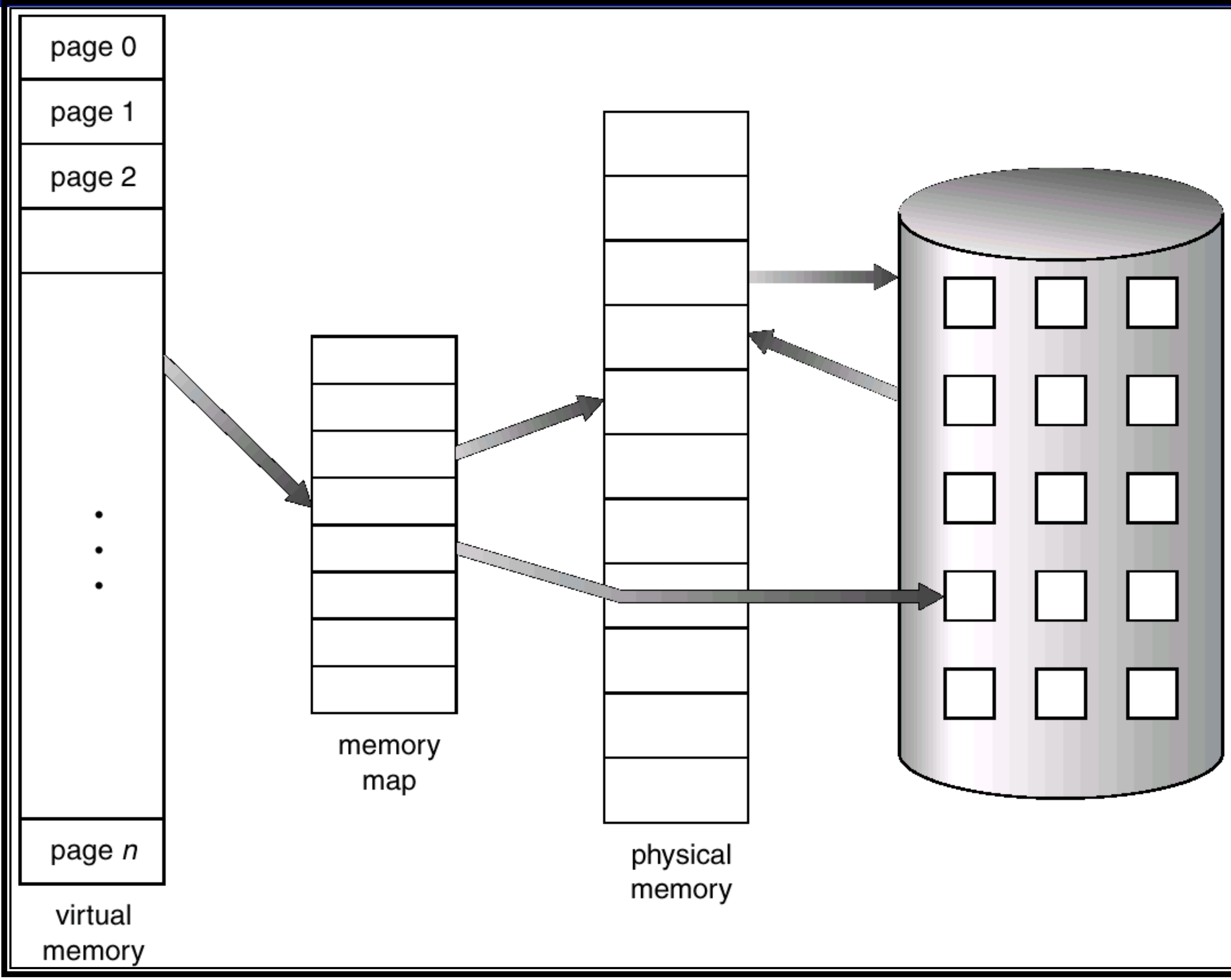
Outline

- Chapter 10: Virtual Memory
  - Demand Paging
  - Process Creation
  - Page Replacement
  - Allocation of Frames
  - Thrashing
- The robustness of NUMA Memory Management Richard P. LaRowe Jr., Carla Schlatter Ellis and Laurence S. Kaplan 13th SOSP
- Practical, Transparent Operating System Support for Superpages Juan Navarro, Sitaram Iyer, Peter Druschel, and Alan Cox , In Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002)

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 (trap 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

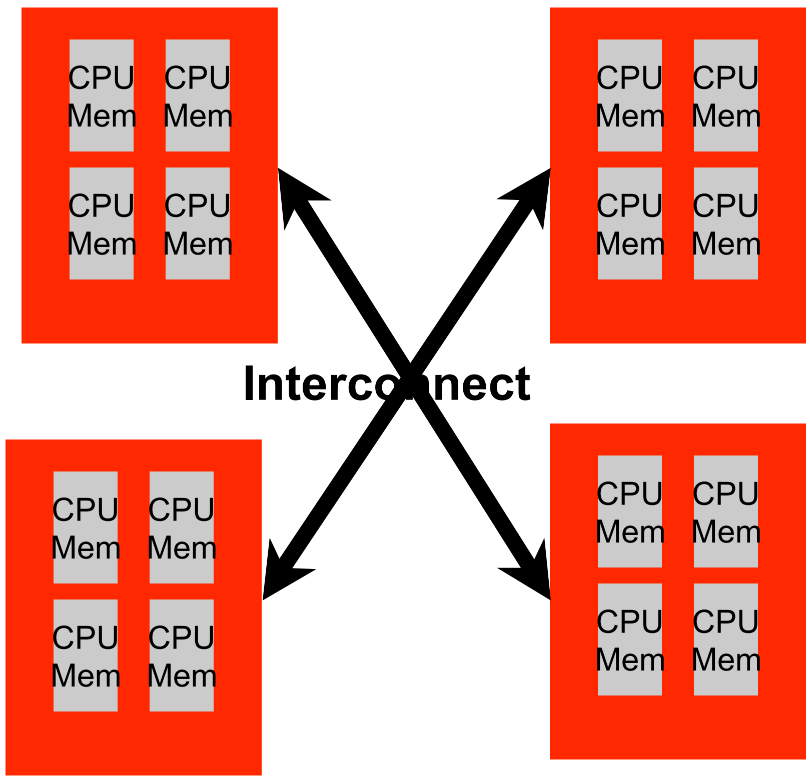
- Replace the page that will not be used for the longest period of time
- Approximations
  - FIFO - Belady's anamoly
  - Least recently used (LRU)
    - Counters: Update every page access
    - Stack of page numbers used
    - Additional reference bit algorithm
    - Second chance algorithm
    - Enhanced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

Page rates

- Paging rates affected by code (compiler)
- Locking memory for IO completion

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

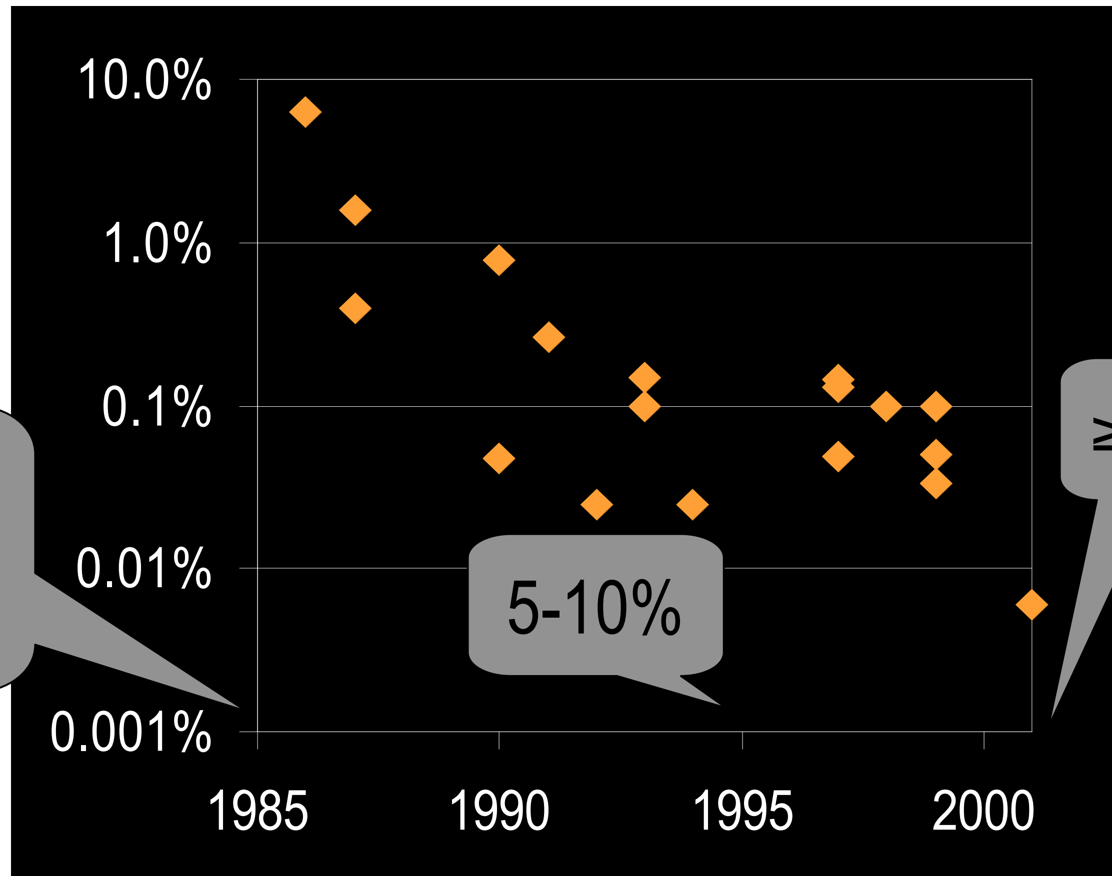
Superpages

- Increasing cost in TLB miss overhead
  - growing working sets
  - TLB size does not grow at same pace
- Processors now provide superpages
  - one TLB entry can map a large region
- OSs have been slow to harness them
  - no transparent superpage support for apps

TLB coverage trend

TLB coverage as percentage of main memory

Factor of 1000 decrease in 15 years

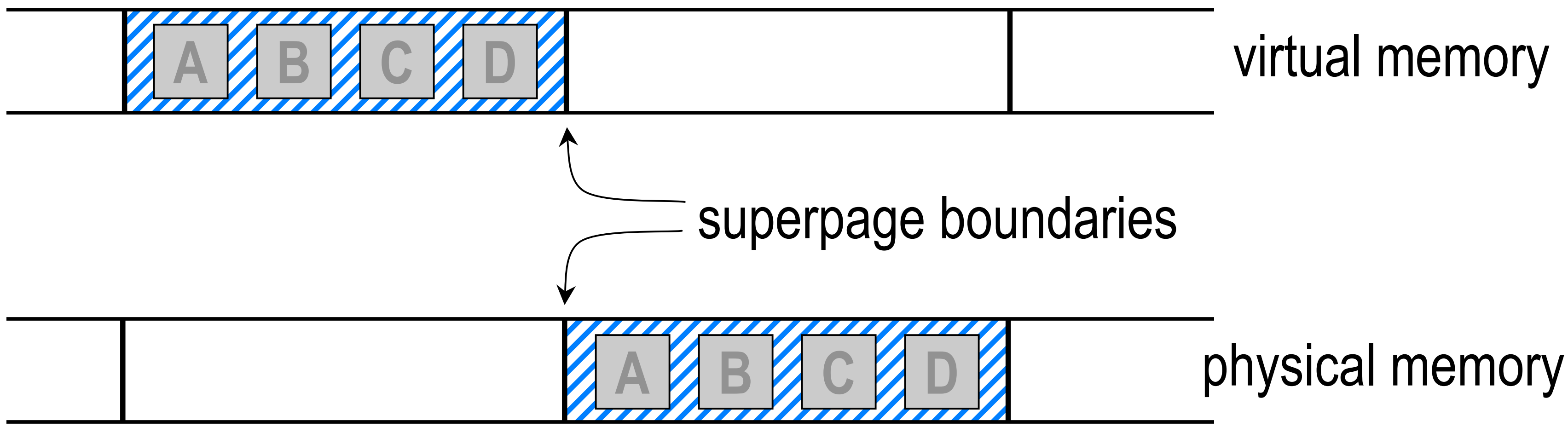


Why multiple superpage sizes

- Improvements with only one superpage size vs. all sizes

	64KB	512KB	4MB	All
FFT	1%	0%	**55%**	55%
galgel	**28%**	**28%**	1%	29%
mcf	24%	31%	22%	**68%**

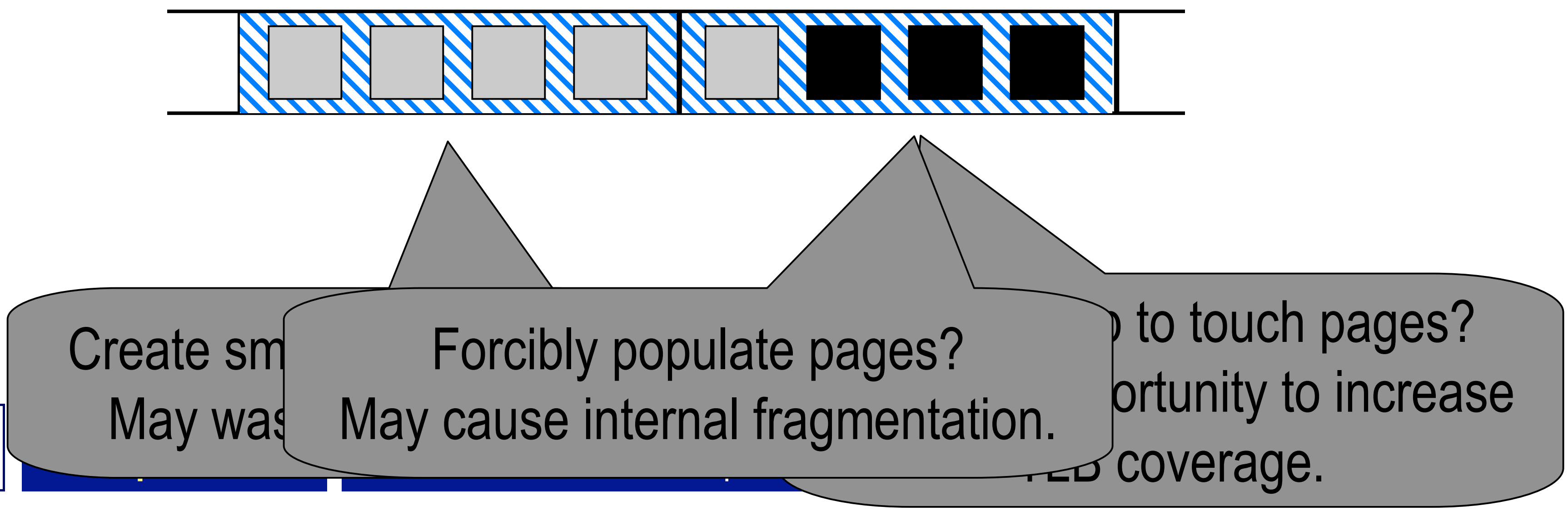
Issue 1: superpage allocation



- How / when / what size to allocate?

Issue 2: promotion

- Promotion: create a superpage out of a set of smaller pages
  - mark page table entry of each base page
- When to promote?



Issue 3: demotion

- Demotion: convert a superpage into smaller pages
- when page attributes of base pages of a superpage become non-uniform
- during partial pageouts

Issue 4: fragmentation

- Memory becomes fragmented due to
  - use of multiple page sizes
  - persistence of file cache pages
  - scattered wired (non-pageable) pages
- Contiguity: contended resource
- OS must
  - use contiguity restoration techniques
  - trade off impact of contiguity restoration against superpage benefits