Operating Systems
Virtual Memory

Lecture 11

Michael O'Boyle

Paged virtual memory

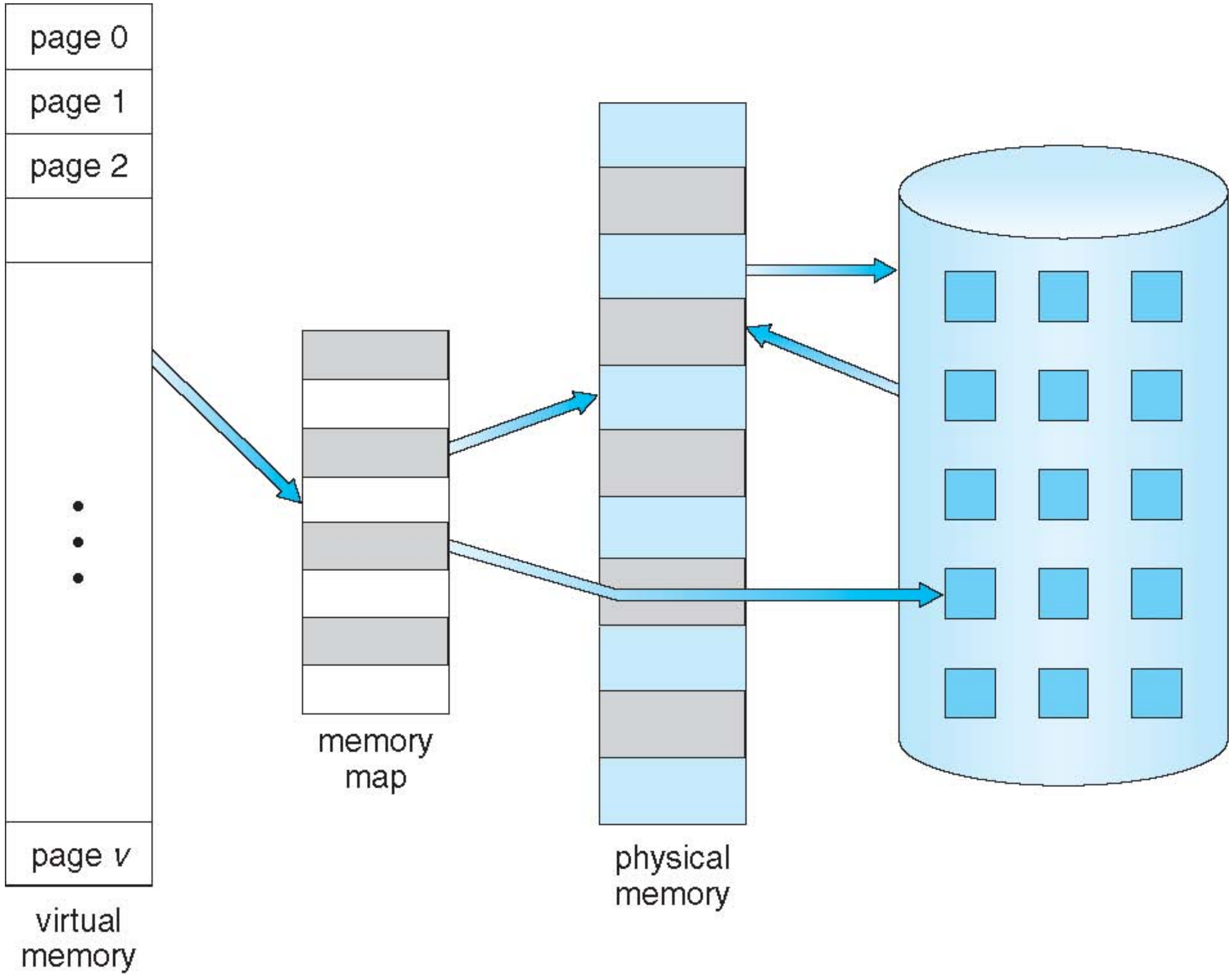
Allows a larger logical address space than physical memory

All pages of address space do not need to be in memory

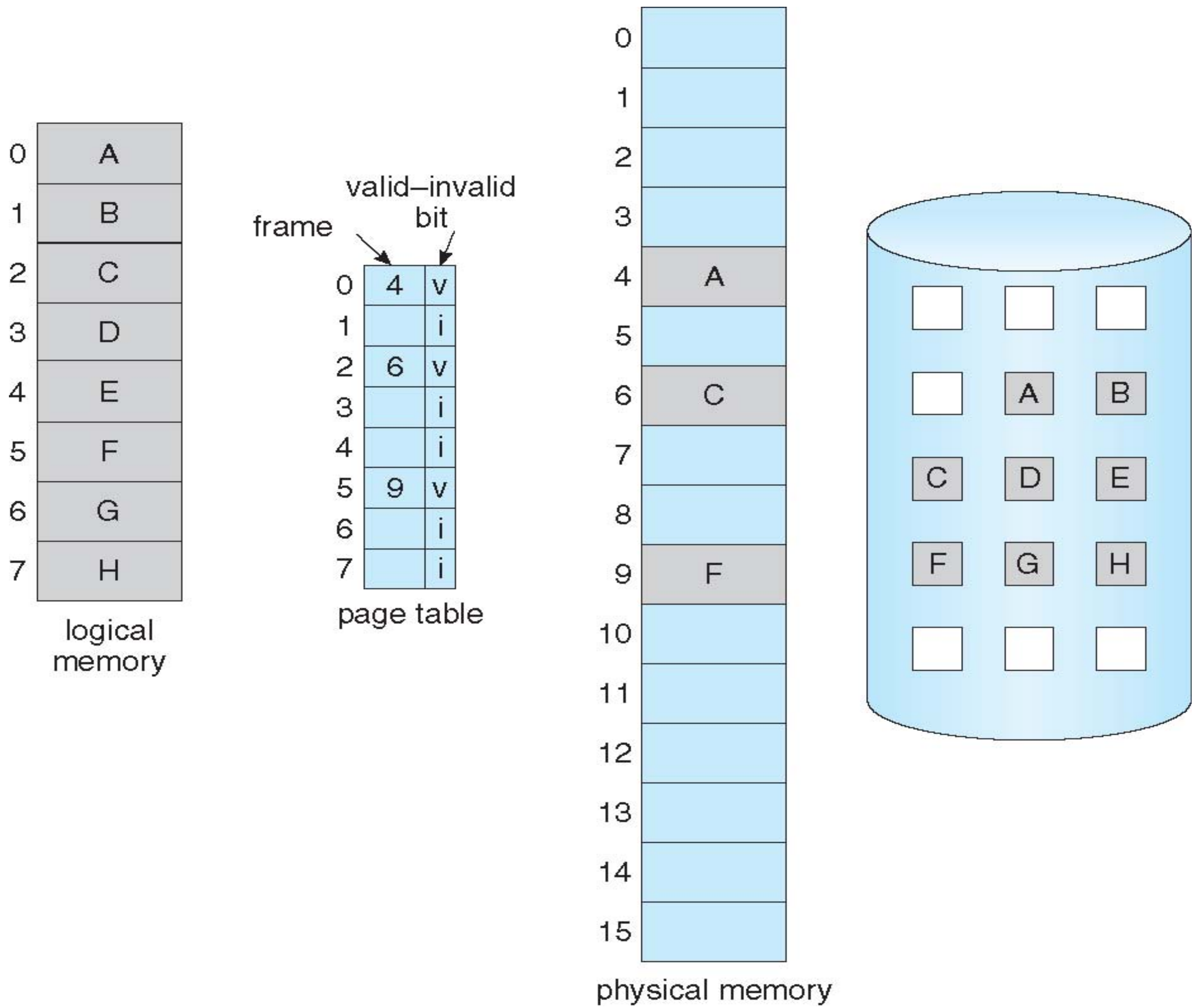
- the full (used) address space on disk in page-sized blocks
- main memory used as a (page) cache

- Needed page transferred to a free page frame
  - if no free page frames, evict a page
    - evicted pages go to disk only if dirty
  - Transparent to the application, except for performance
  - managed by hardware and OS
- Traditionally called paged virtual memory

Virtual Memory That is Larger Than Physical Memory



Page Table When Some Pages Are Not in Main Memory



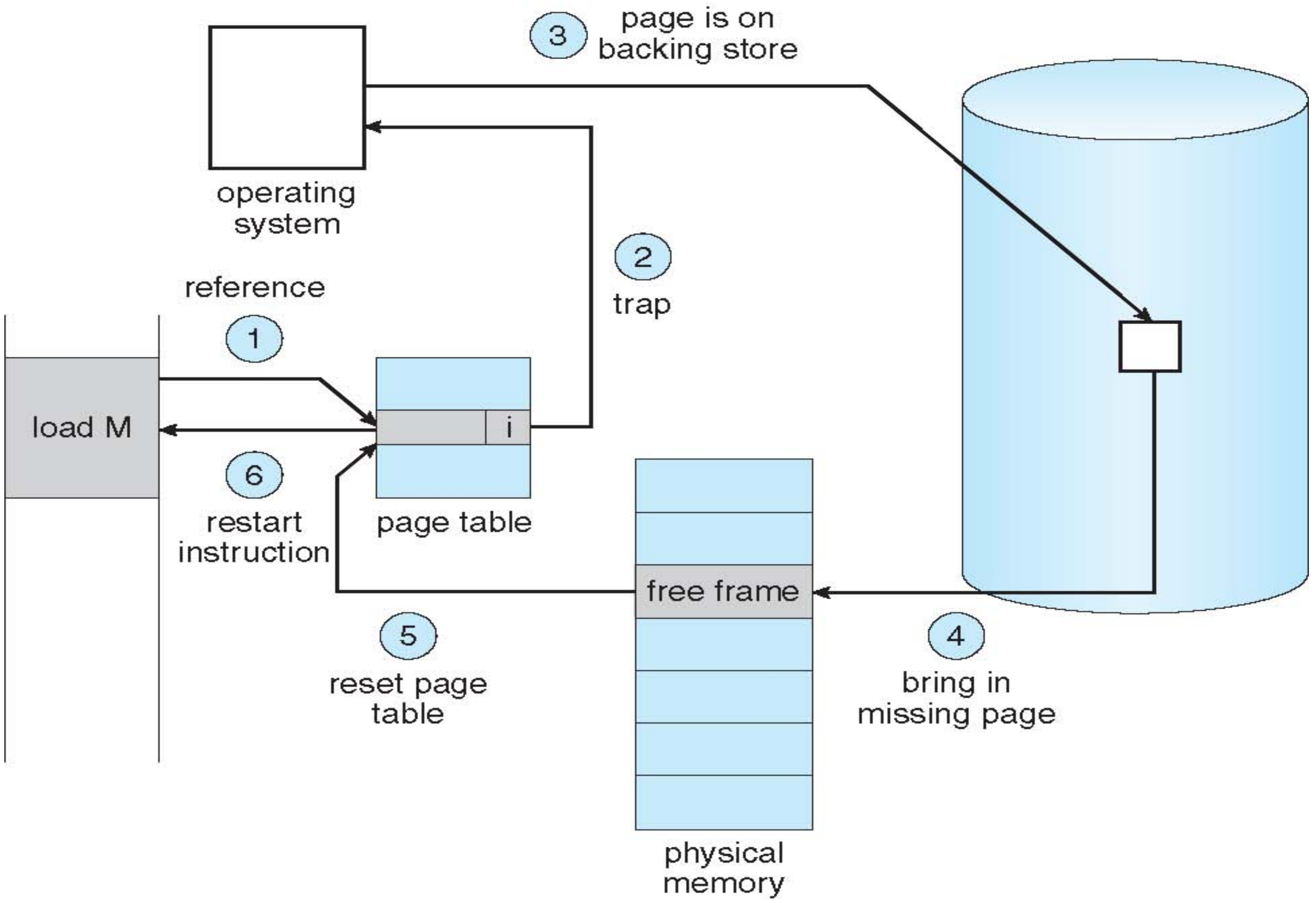
Page Fault

- If there is a reference to a page, first reference to that page will trap to operating system:

  page fault

1. Operating system looks at another table to decide:
   - Invalid reference ⇒ abort
   - Just not in memory
2. Find free frame
3. Swap page into frame via scheduled disk operation
4. Reset tables to indicate page now in memory
   Set validation bit = **v**
5. Restart the instruction that caused the page fault

Steps in Handling a Page Fault



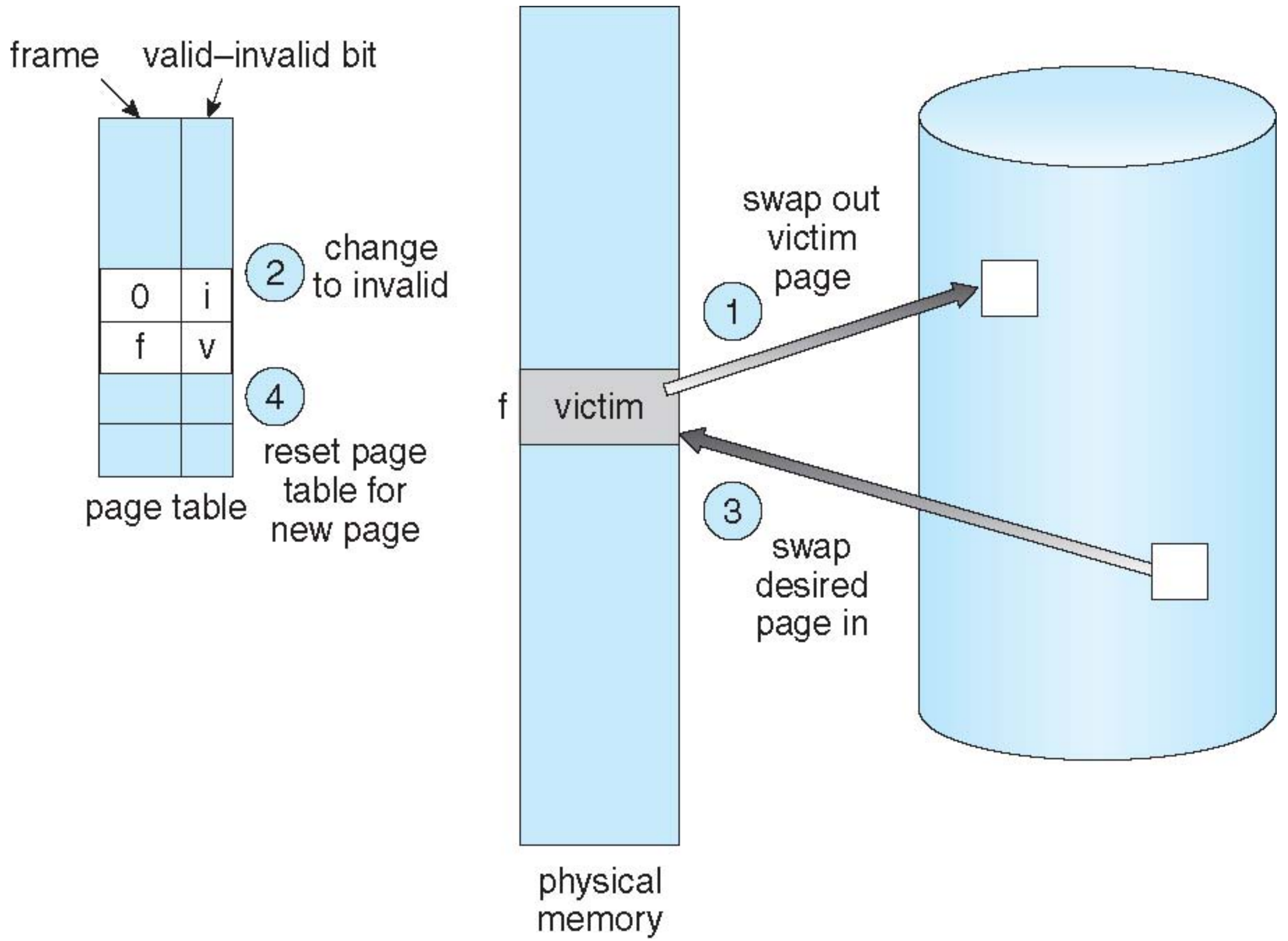
Demand paging

- Pages only brought into memory when referenced
  - Only code/data that is needed by a process needs to be loaded
    - What's needed changes over time
  - Hence, it's called demand paging
- Few systems try to anticipate future needs
- But sometimes cluster pages
  - OS keeps track of pages that should come and go together
  - bring in all when one is referenced
  - interface may allow programmer or compiler to identify clusters

Page replacement

- When you read in a page, where does it go?
  - if there are free page frames, grab one
  - if not, must evict something else
    - this is called page replacement
- Page replacement algorithms
  - try to pick a page that won't be needed in the near future
  - try to pick a page that hasn't been modified (thus saving the disk write)
- OS tries to keep a pool of free pages around
  - so that allocations don't inevitably cause evictions
- OS tries to keep some "clean" pages around
  - so that even if you have to evict a page, you won't have to write it

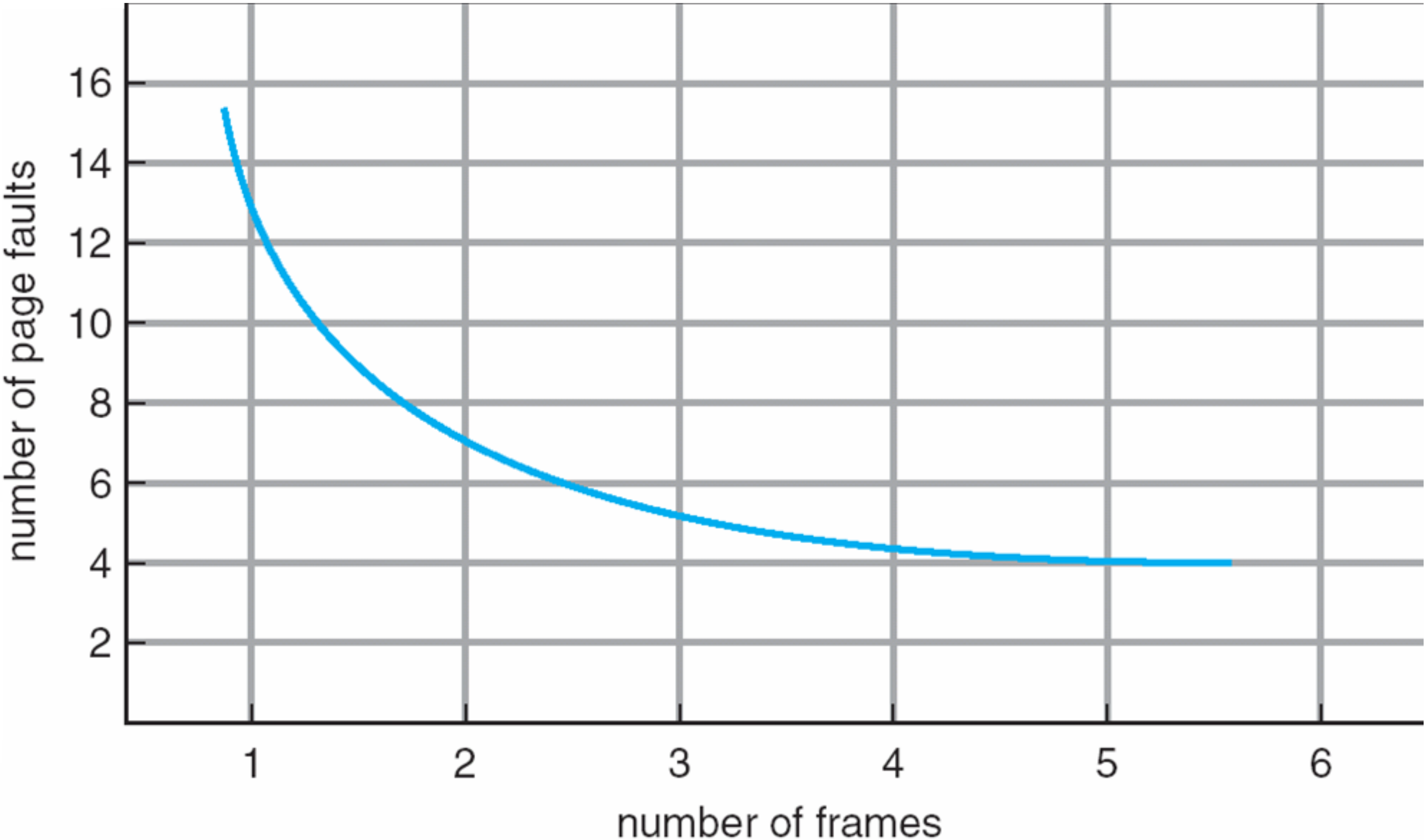
Page Replacement



Evicting the best page

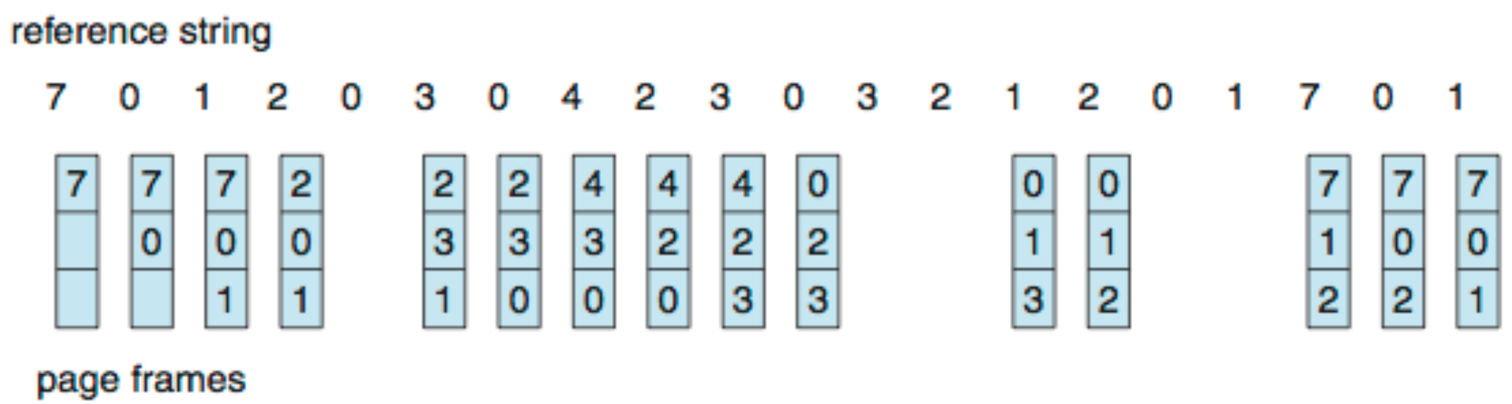
- The goal of the page replacement algorithm:
  - reduce fault rate by selecting best victim page to remove
  - the best page to evict is one that will never be touched again
  - Belady's proof:
    - evicting the page that won't be used for the longest period of time minimizes page fault rate
- Examine page replacement algorithms
  - assume that a process pages against itself
  - using a fixed number of page frames
- Number of frames available impacts page fault rate
  - Note Belady's anomaly

Graph of Page Faults Versus The Number of Frames



First-In-First-Out (FIFO) Algorithm

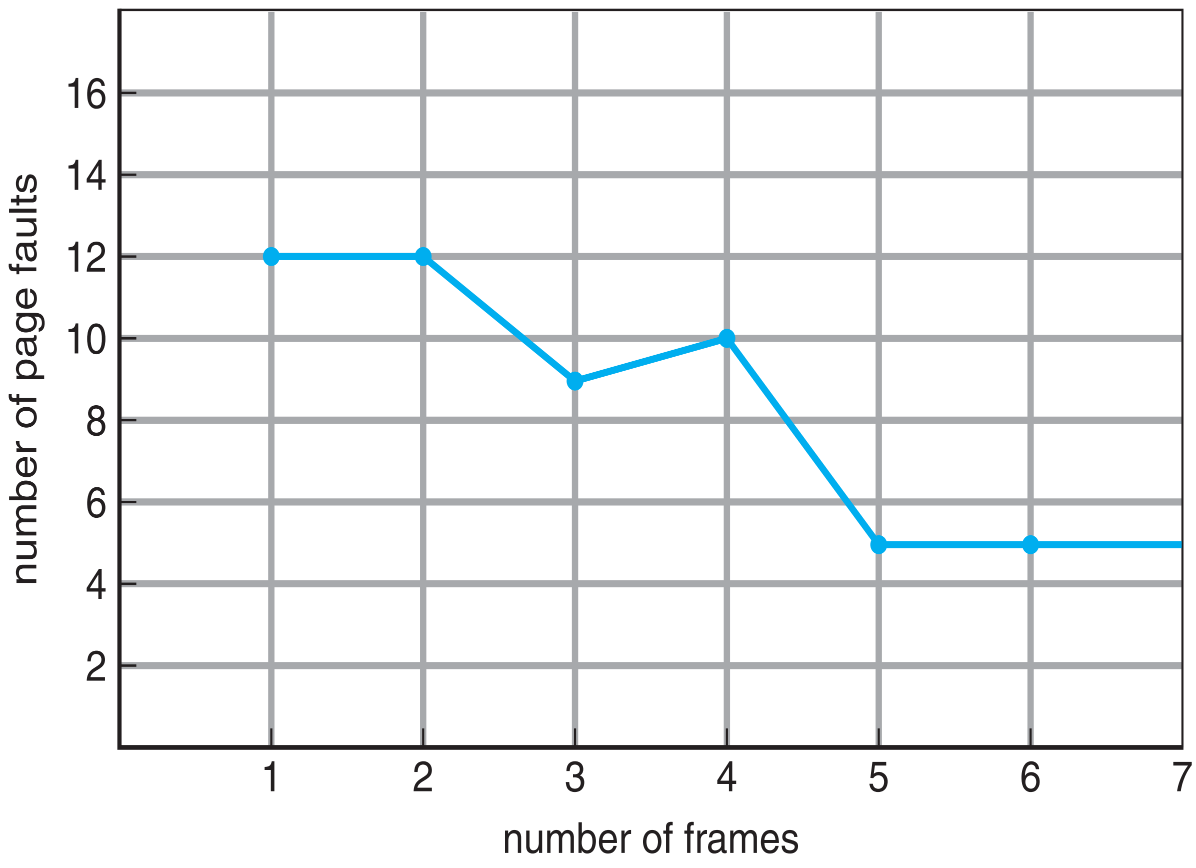
- Reference string: **7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1**
- 3 frames (3 pages can be in memory at a time per process)



15 page faults

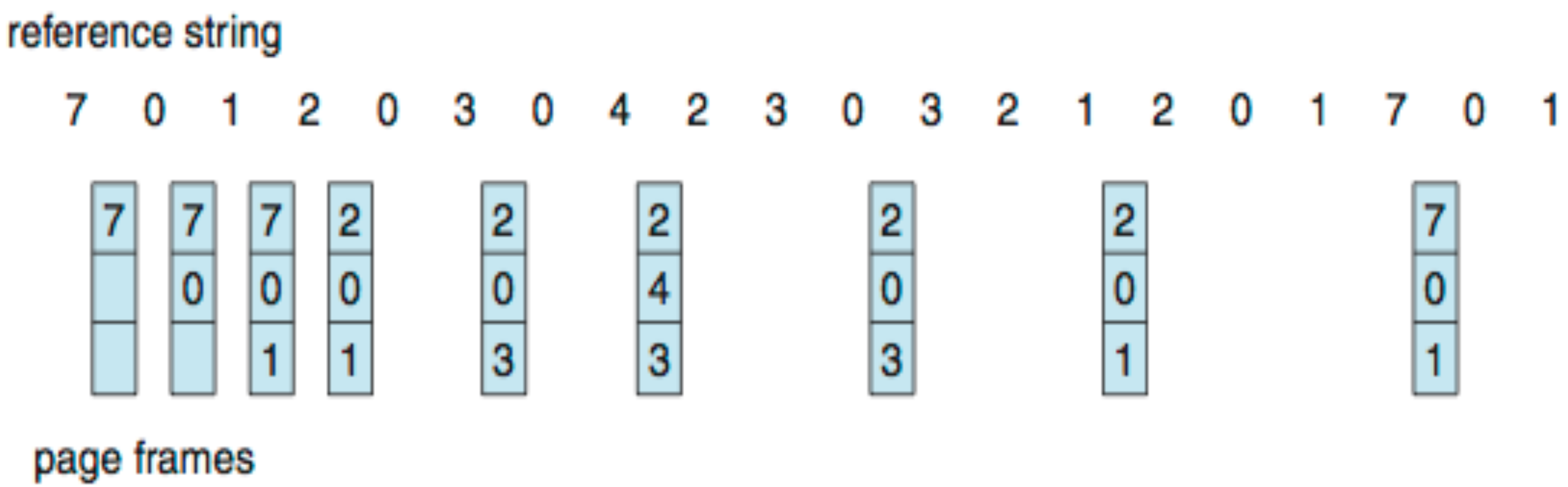
- Can vary by reference string: consider 1,2,3,4,1,2,5,1,2,3,4,5
  - Adding more frames can cause more page faults!
    - **Belady's Anomaly**

FIFO Illustrating Belady's Anomaly



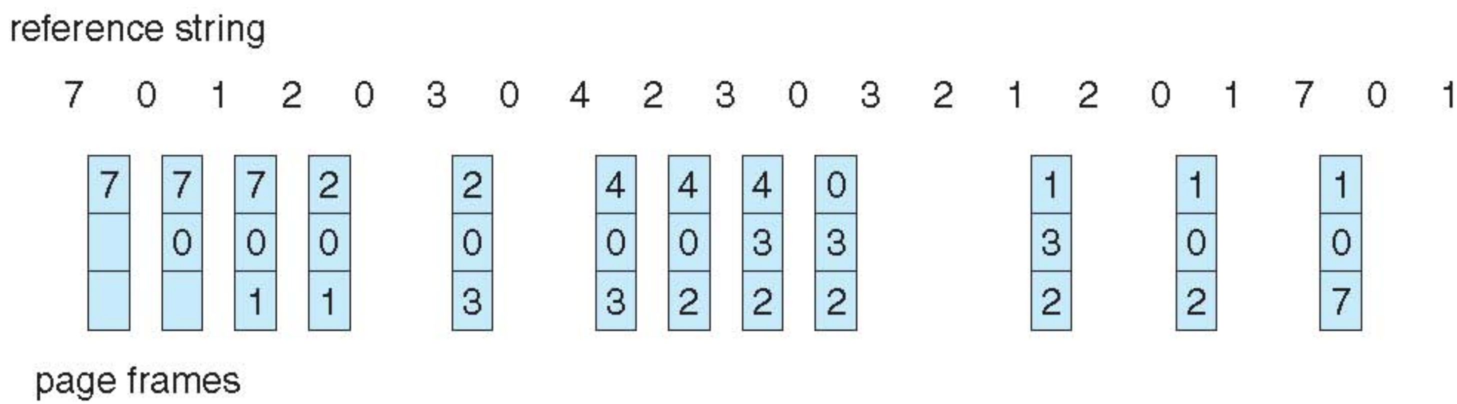
Belady’s Optimal Algorithm

- Replace page that will not be used for longest period of time
  - 9 is optimal for the example
- How do you know this?
  - Can’t read the future
- Used for measuring how well your algorithm performs



Least Recently Used (LRU) Algorithm

- Use past knowledge rather than future
- Replace page that has not been used in the most amount of time
- Associate time of last use with each page



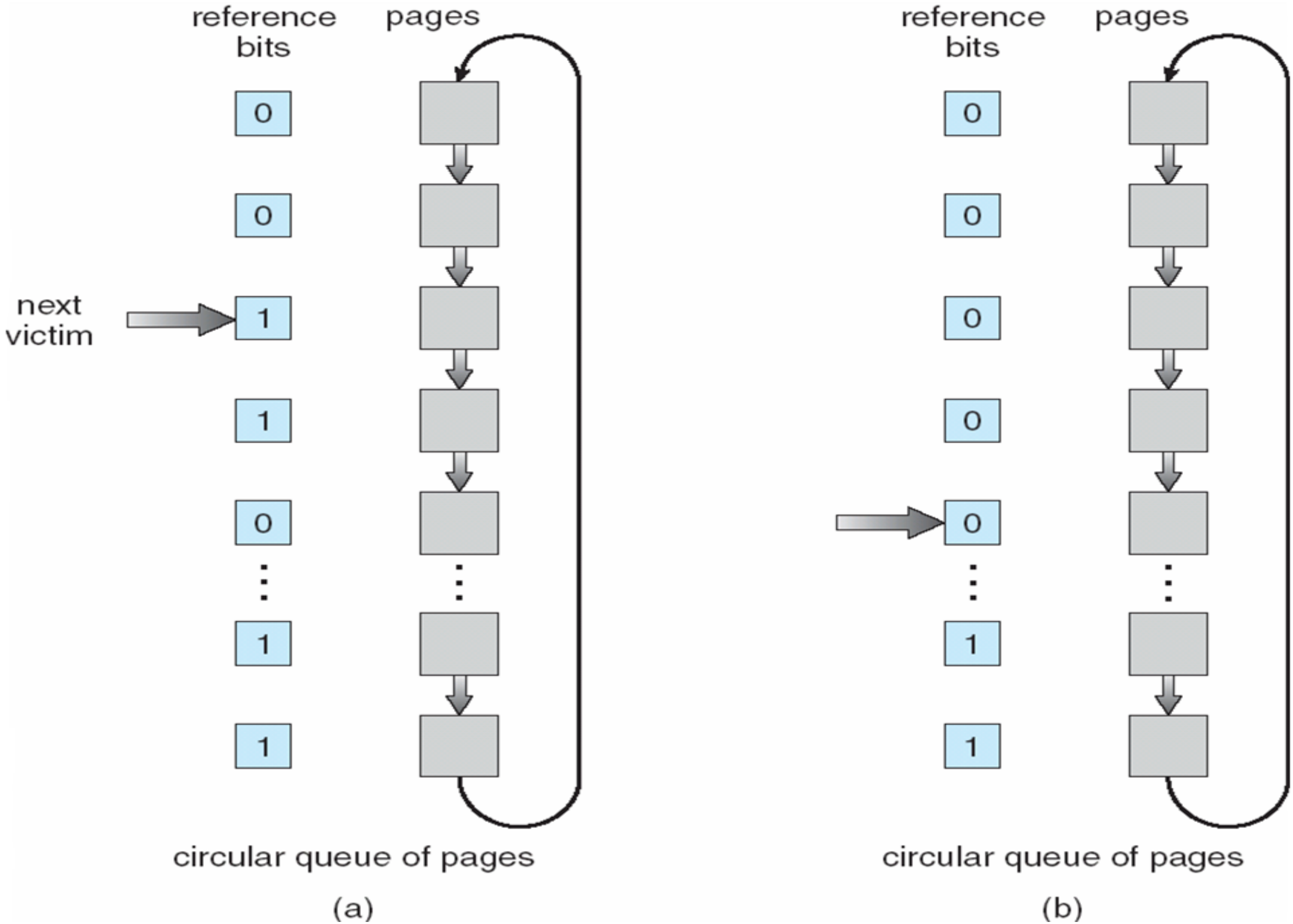
- 12 faults – better than FIFO but worse than Belady's/ OPT
- Generally good algorithm and frequently used
- But how to implement?

Approximating LRU

- Many approximations, all use the PTE's referenced bit
  - keep a counter for each page
  - at some regular interval, for each page, do:
    - if ref bit = 0, increment the counter (hasn't been used)
    - if ref bit = 1, zero the counter (has been used)
    - regardless, zero ref bit
  - the counter will contain the # of intervals since the last reference to the page
    - page with largest counter is least recently used
- Some architectures don't have PTE reference bits
  - can simulate reference bit using the valid bit to induce faults

Second-chance Clock

- Not Recently Used (NRU) or Second Chance
  - replace page that is “old enough”
  - logically, arrange all physical page frames in a big circle (clock)
    - just a circular linked list
- A “clock hand” is used to select a good LRU candidate
    - sweep through the pages in circular order like a clock
- If ref bit is off, it hasn’t been used recently, we have a victim
- If the ref bit is on, turn it off and go to next page
  - arm moves quickly when pages are needed

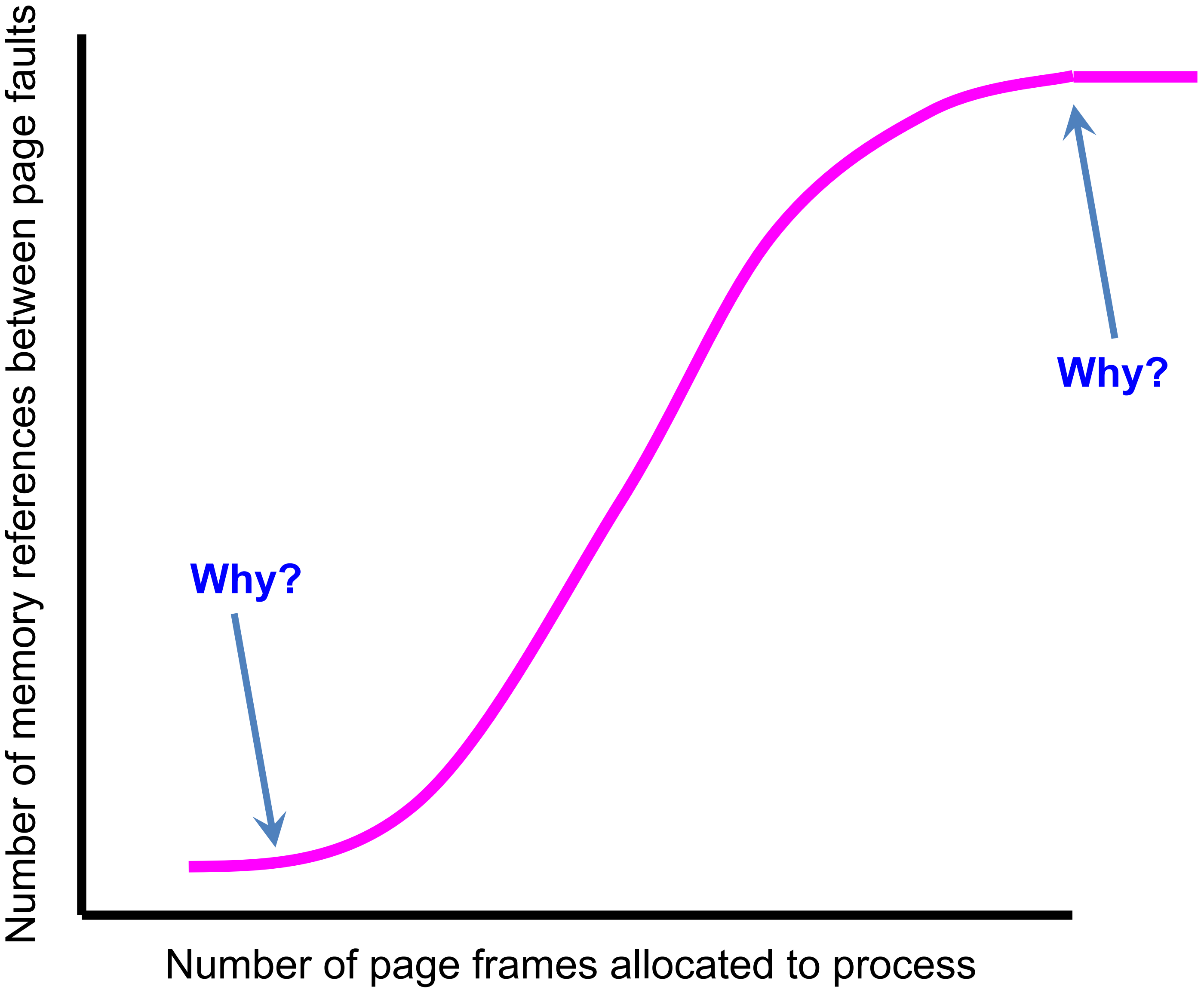

reference bits
pages
next victim
0
0
1
1
0
1
1
circular queue of pages
(a)
reference bits
pages
0
0
0
0
0
1
1
circular queue of pages
(b)

Allocation of frames among processes

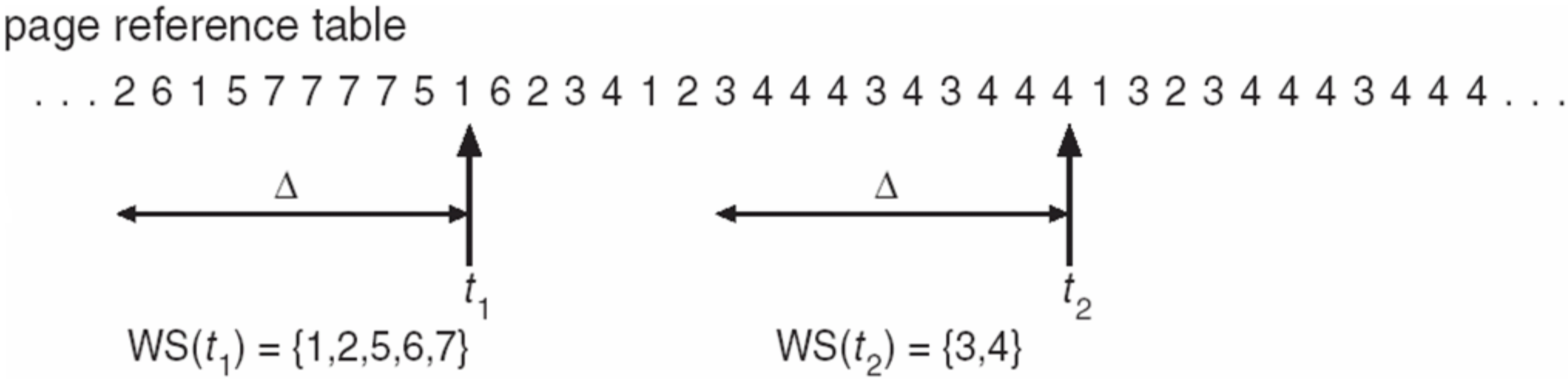
- FIFO and LRU Clock each can be implemented as either local or global replacement algorithms
  - local
    - each process is given a limit of pages it can use
    - it "pages against itself" (evicts its own pages)
  - global
    - the "victim" is chosen from among all page frames, regardless of owner
    - processes' page frame allocation can vary dynamically
- Issues with local replacement?
  - poor utilization of free page frames, long access time
- Issues with global replacement?
  - Linux uses global replacement: global thrashing

The *working set model of program behavior*

- Working set of a process is used to model the dynamic locality of its memory usage
  - working set = set of pages process currently “needs”
  - formally defined by Peter Denning in the 1960’s
- Definition:
  - WS(t,w) = {pages P such that P was referenced in the time interval (t, t-w)}
    - t: time
    - w: working set *window* (measured in page refs)
    - a page in WS only if it was referenced in the last w references
- Working set varies over the life of the program
  - so does the working set size


Number of memory references between page faults
Number of page frames allocated to process
Why?
Why?

Example: Working set



Working set size

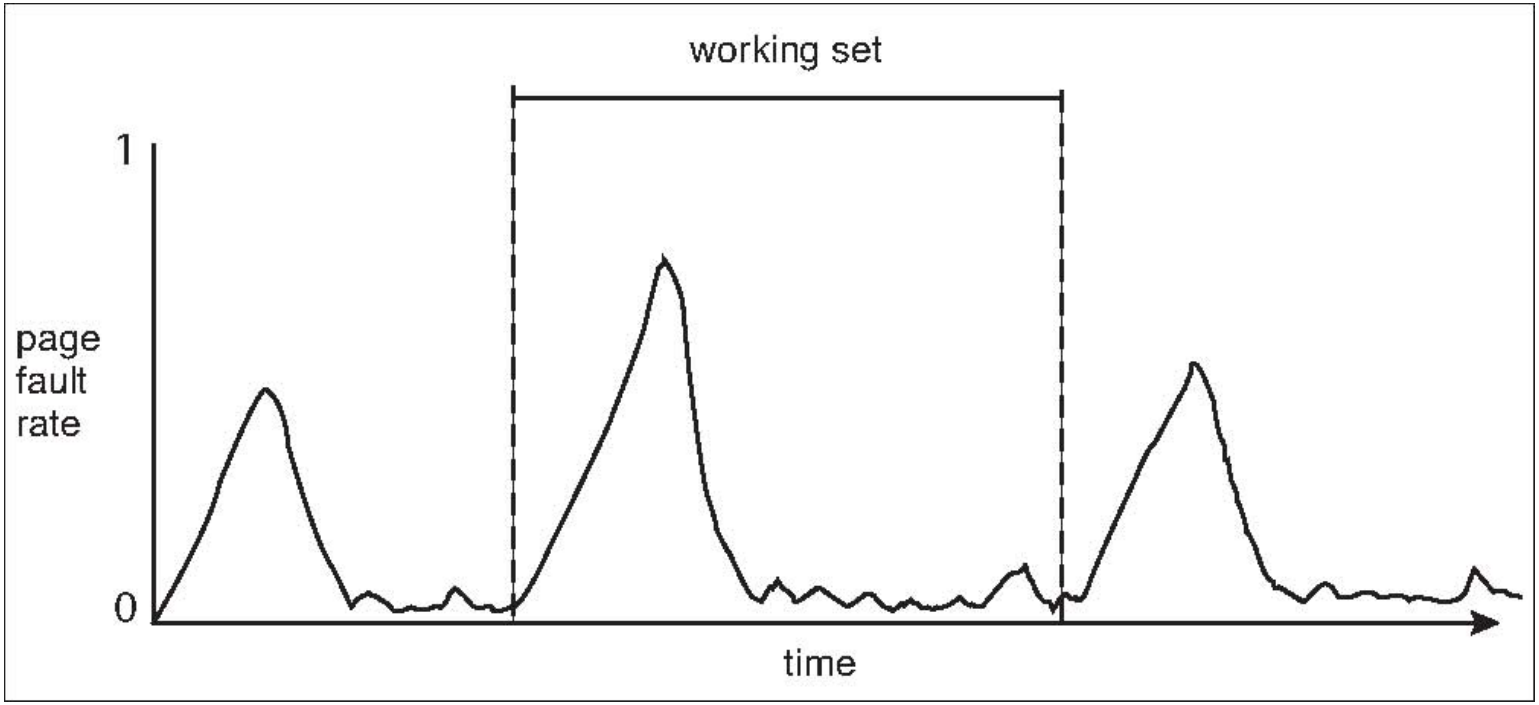
- The working set size, |WS(t,w)|,
  - Changes with program locality
- During periods of poor locality,
  - more pages are referenced
- Within that period of time,
  - the working set size is larger
- Intuitively, the working set must be in memory,
  - otherwise you'll experience heavy faulting
  - thrashing

Hypothetical Working Set algorithm

- Estimate |WS(0,w)| for a process
  - Allow that process to start only if you can allocate it that many page frames
- Use a local replacement algorithm (LRU Clock?)
  - make sure that the working set are occupying the process's frames
- Track each process's working set size,
  - and re-allocate page frames among processes dynamically
- Problem
  - keep track of working set size.
- Use reference bit with a fixed-interval timer interrupt

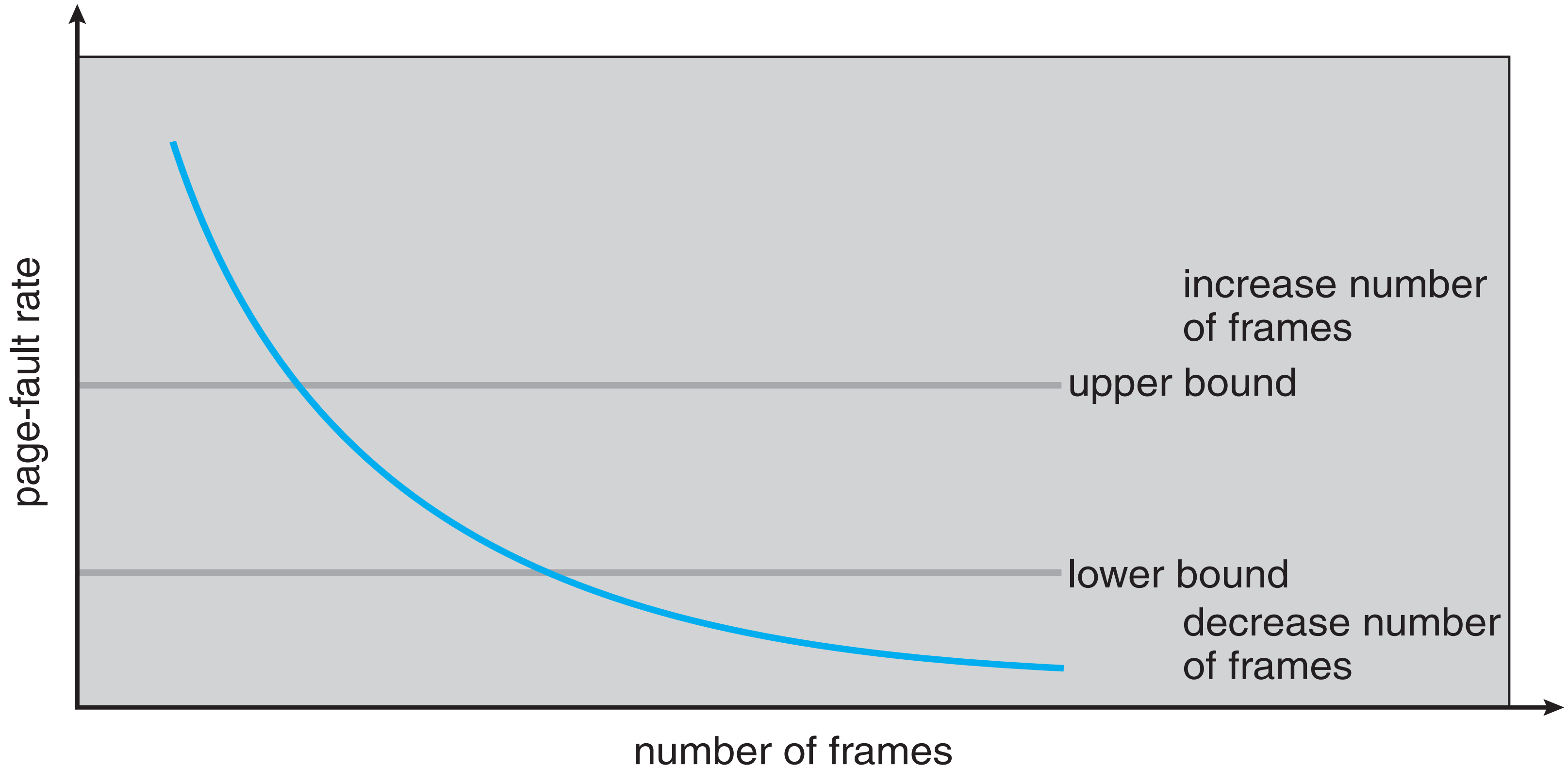
Working Sets and Page Fault Rates

- Direct relationship between working set of a process and its page-fault rate
- Working set changes over time
- Peaks and valleys over time



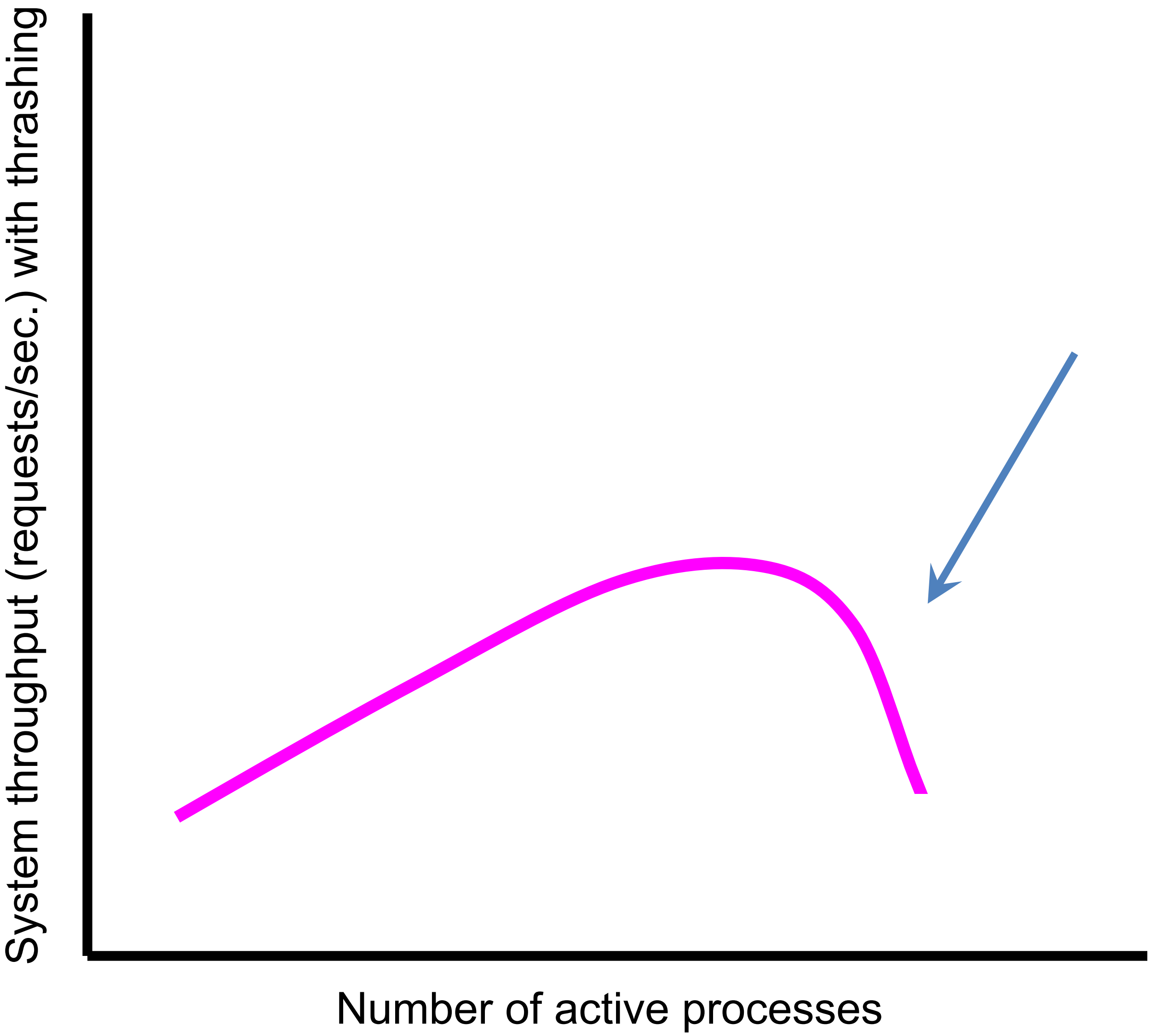
Page-Fault Frequency

- More direct approach than WSS
- Establish "acceptable" **page-fault frequency** (**PFF**) rate and use local replacement policy
  - If actual rate too low, process loses frame
  - If actual rate too high, process gains frame



Thrashing

- Thrashing
  - when the system spends most of its time servicing page faults, little time doing useful work
- Could be that there is enough memory
  - but a poor replacement algorithm - incompatible with program behavior
- Could be that memory is over-committed
  - OS sees CPU poorly utilized and adds more processes
    - too many active processes
  - Makes problem worse


System throughput (requests/sec.) with thrashing
Number of active processes

Summary

- Virtual memory
- Page faults
- Demand paging
  - don’t try to anticipate
- Page replacement
  - Belady, LRU, Clock,
  - local, global
- Locality
  - temporal, spatial
- Working set
- Thrashing