IOWA STATE UNIVERSITY
Department of Electrical and Computer Engineering

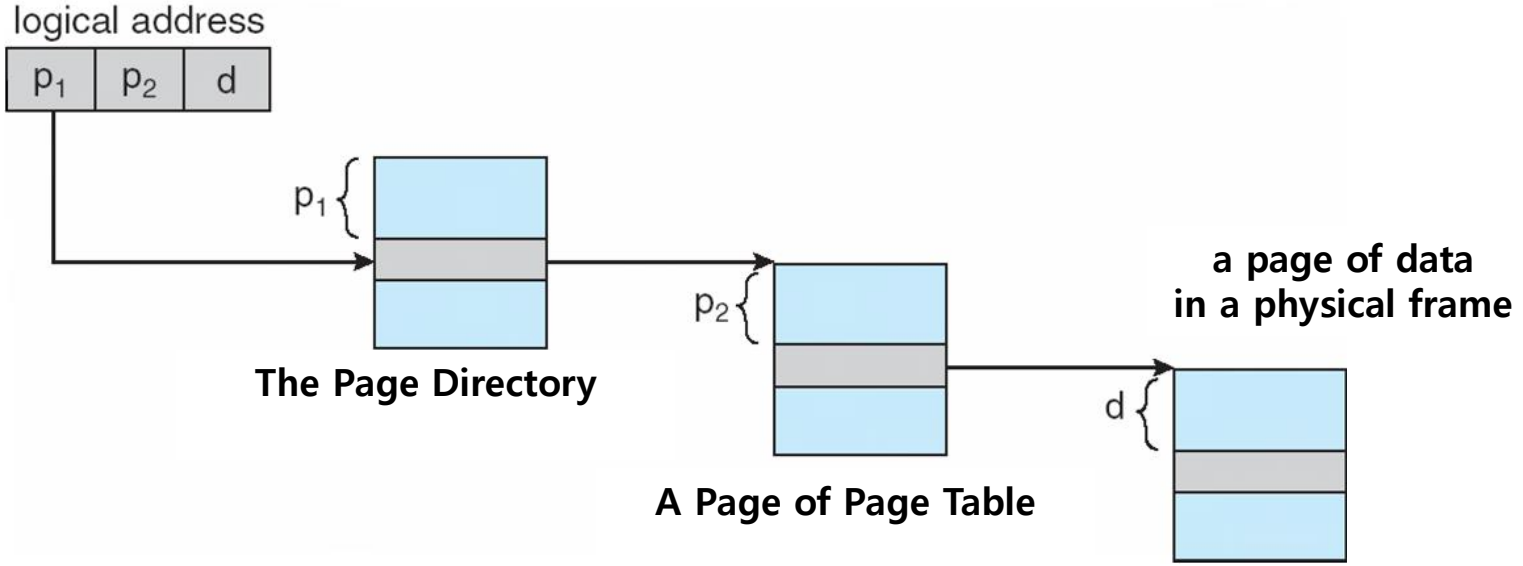
Lecture 24:
Page Replacement Algorithms

Agenda

- Recap
- Page Replacement Algorithms
  - FIFO (First-In-First-Out)
  - LRU (Least-Recently-Used)
  - Clock

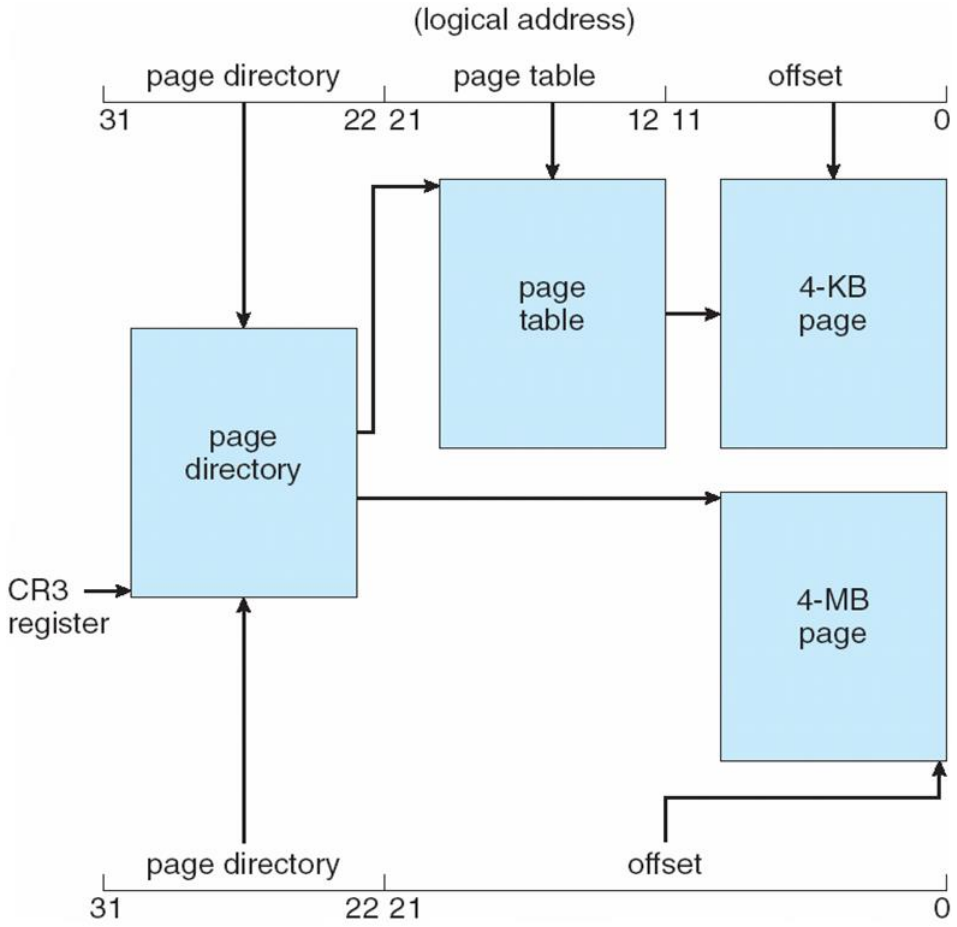
Recap

- Multi-Level Page Tables
  - Paging the page table itself
    - e.g., a two-level page table
      - the page directory index (p1) is used to identify a page directory entry (PDE) in the page directory
      - the page table index (p2) is used to identify a page table entry



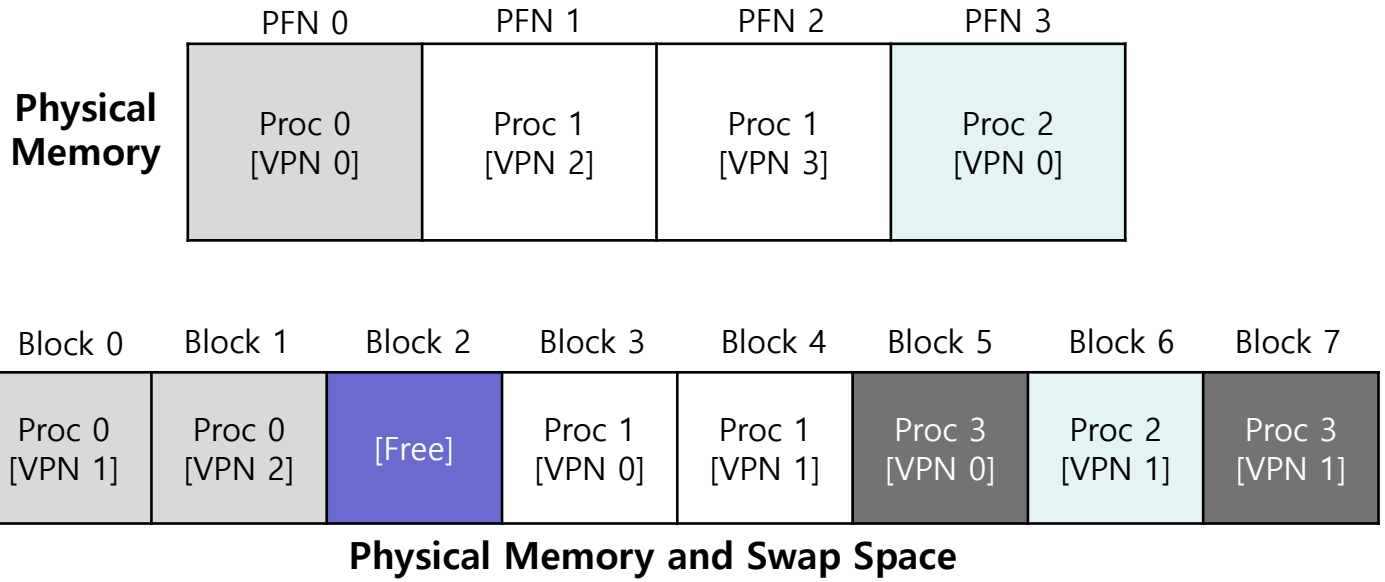
Recap

- Intel IA-32 Paging Architecture



Recap

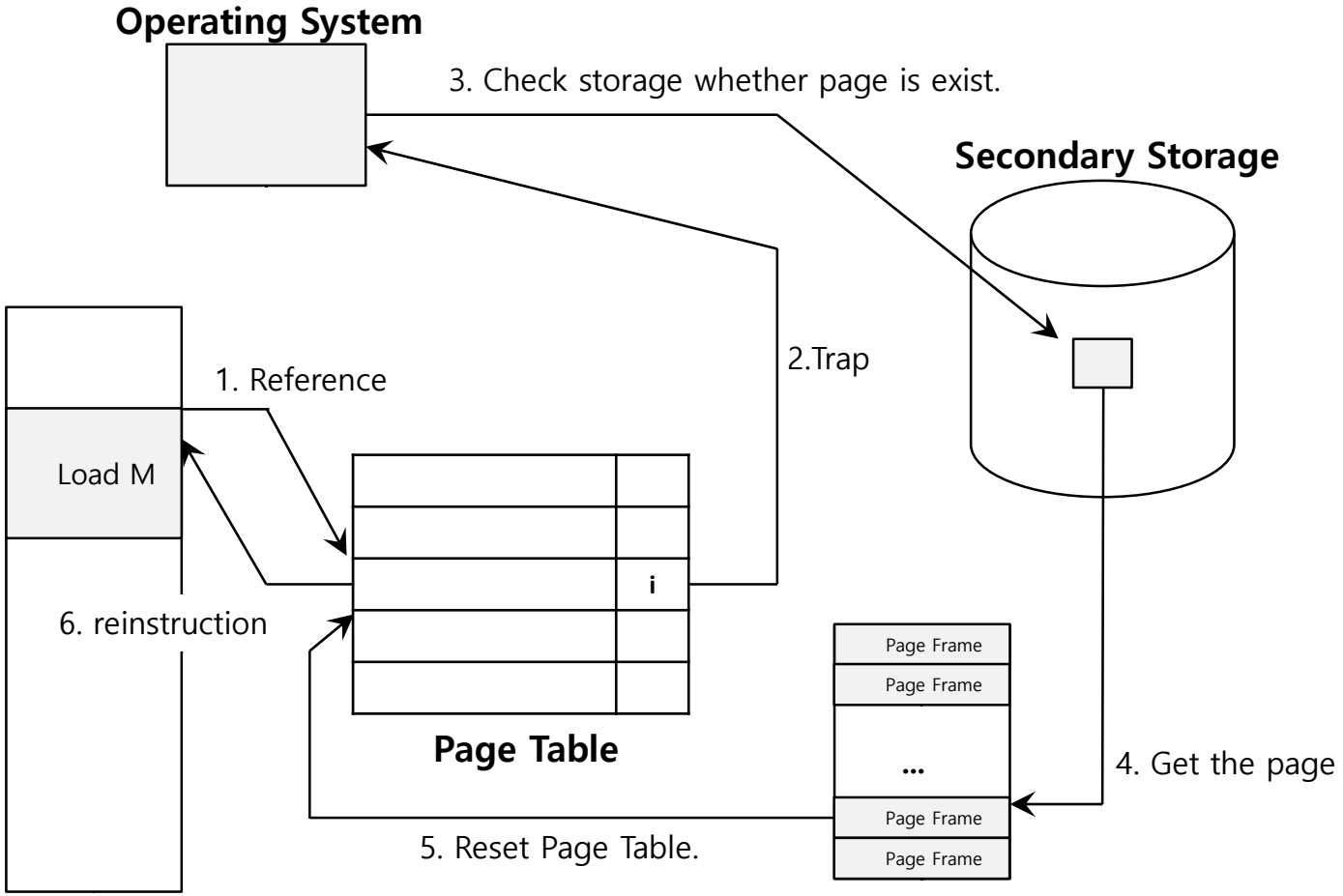
- Swapping
  - stash away portions of address space that currently aren't in great demand
  - the unpopular pages are placed in the **swap space** at the next layer of memory/storage hierarchy



Physical Memory and Swap Space

Recap

- Swap in a page from disk at page fault



Agenda

- ~~Recap~~
- Page Replacement Algorithms
  - FIFO (First-In-First-Out)
  - LRU (Least-Recently-Used)
  - Clock

Page Replacement Algorithms

- When free space in physical memory is low, OS starts evicting pages to disks
- Page replacement algorithms decide which page to evict
  - the physical memory serves as a cache of the swap space
    - a cache miss leads to a page fault
  - the goal is to minimize the number of cache misses/page faults
    - i.e., minimize the accesses to the swap space

The Optimal Algorithm

- Lead to the fewest number of misses overall
  - Replace the page that will be accessed furthest in the future
  - Resulting in the fewest-possible cache misses
    - or the highest **hit rate**
- Need future access information
  - usually unknown in practice
  - serve only as a comparison point, to see how close we are to the (theoretical) best case scenario

The Optimal Algorithm

- Example

References

0 1 2 0 1 3 0 3 1 2 1

Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		0
1	Miss		0,1
2	Miss		0,1,2
0	Hit		0,1,2
1	Hit		0,1,2
3	Miss	2	0,1,3
0	Hit		0,1,3
3	Hit		0,1,3
1	Hit		0,1,3
2	Miss	3	0,1,2
1	Hit		0,1,2

The Optimal Algorithm

- Example

References

0 1 2 0 1 3 0 3 1 2 1

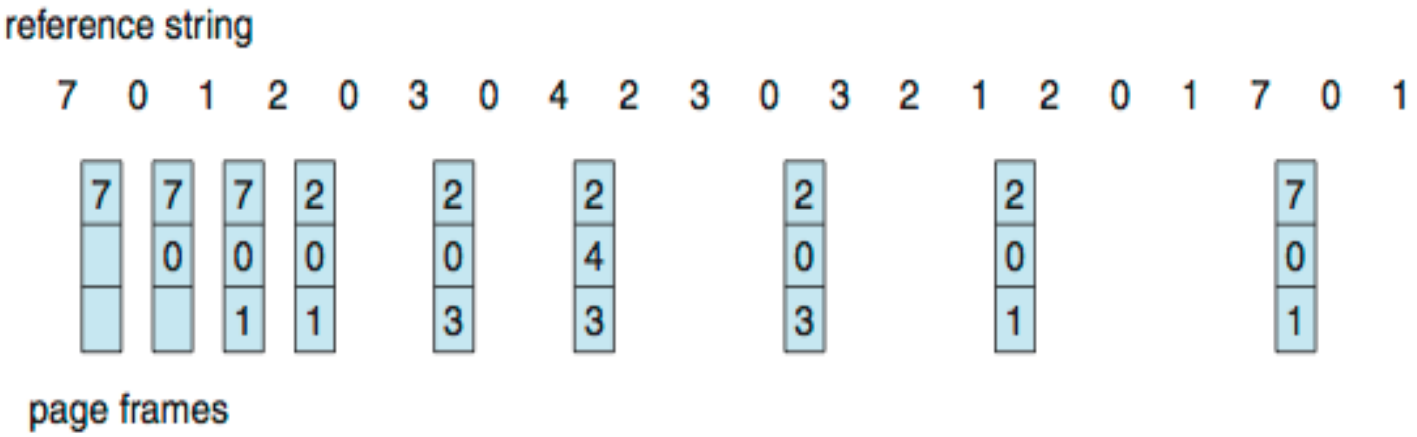
Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		0
1	Miss		0,1
2	Miss		0,1,2
0	Hit		0,1,2
1	Hit		0,1,2
3	Miss	2	0,1,3
0	Hit		0,1,3
3	Hit		0,1,3
1	Hit		0,1,3
2	Miss	3	0,1,2
1	Hit		0,1,2

Hit rate:

$$\frac{Hits}{Hits + Misses} = \mathbf{54.6\%}$$

The Optimal Algorithm

- Another Example
  - 20 references in sequence
  - 3 page frames
  - # of page fault: 9



First In First Out (FIFO) Algorithm

- Pages were recorded in a queue when they enter the system
  - the page on the head of the queue (the "**First-in**" pages) is evicted first
- simple to implement
- irrelevant to the popularity/importance of pages

First In First Out (FIFO) Algorithm

- Example

Reference Row

0 1 2 0 1 3 0 3 1 2 1

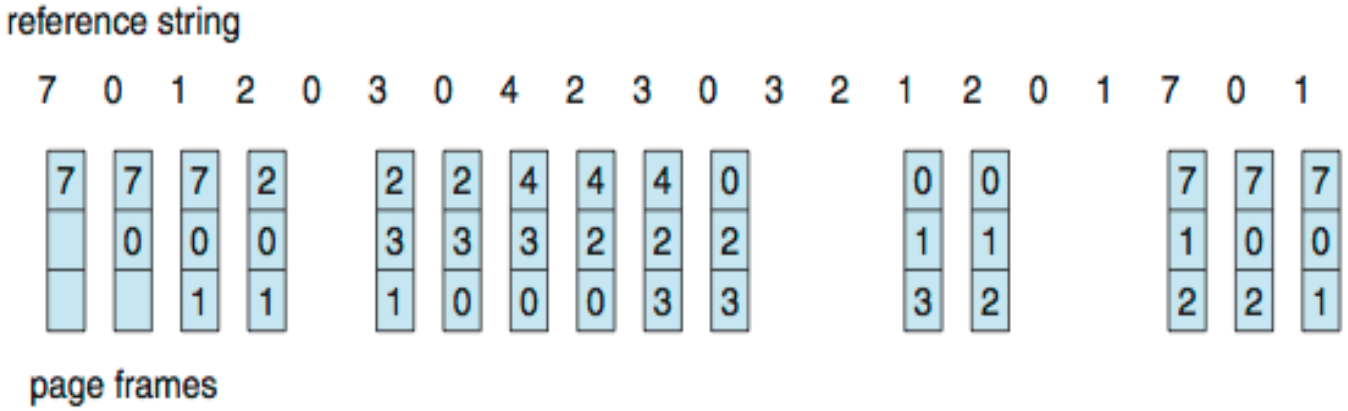
Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		0
1	Miss		0,1
2	Miss		0,1,2
0	Hit		0,1,2
1	Hit		0,1,2
3	Miss	0	1,2,3
0	Miss	1	2,3,0
3	Hit		2,3,0
1	Miss		3,0,1
2	Miss	3	0,1,2
1	Hit		0,1,2

Hit rate:

$$\frac{Hits}{Hits + Misses} = \mathbf{36.4\%}$$

First In First Out (FIFO) Algorithm

- Another Example
  - 20 references in sequence
  - 3 page frames
  - # of page fault: 15



Using History

- Lean on the past and use history.
  - Two type of historical information.

Historical Information	Meaning	Algorithms
recency	The more recently a page has been accessed, the more likely it will be accessed again	LRU (least-recently-used)
frequency	If a page has been accessed many times, It should not be replaced as it clearly has some value	LFU (least-frequently-used)

LRU Algorithm

- Replace the least-recently-used page

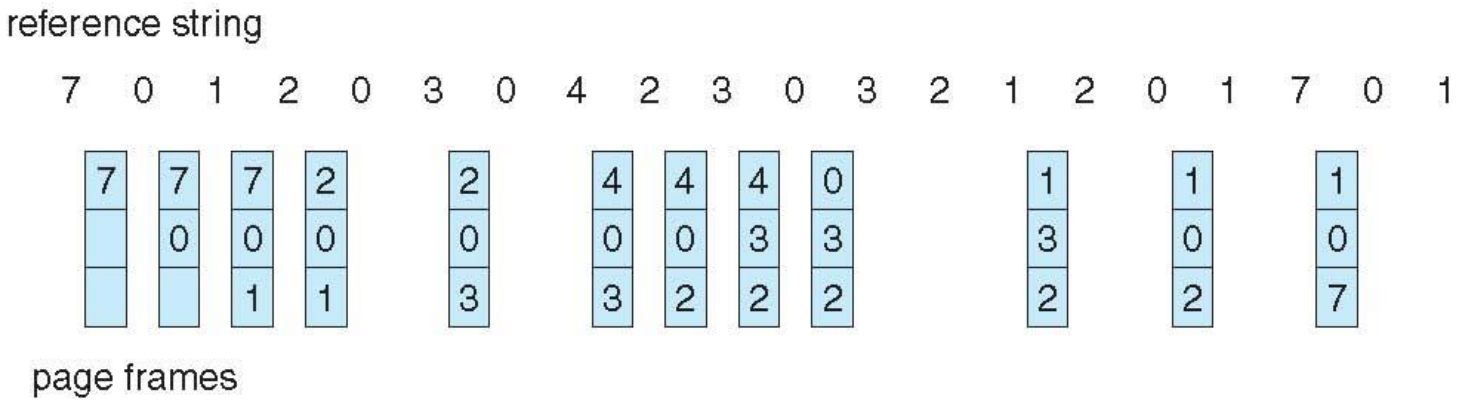
Reference Row

0 1 2 0 1 3 0 3 1 2 1

Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		0
1	Miss		0,1
2	Miss		0,1,2
0	Hit		1,2,0
1	Hit		2,0,1
3	Miss	2	0,1,3
0	Hit		1,3,0
3	Hit		1,0,3
1	Hit		0,3,1
2	Miss	0	3,1,2
1	Hit		3,2,1

LRU Algorithm

- Another Example
  - 20 references in sequence
  - 3 page frames
  - # of page fault: 12

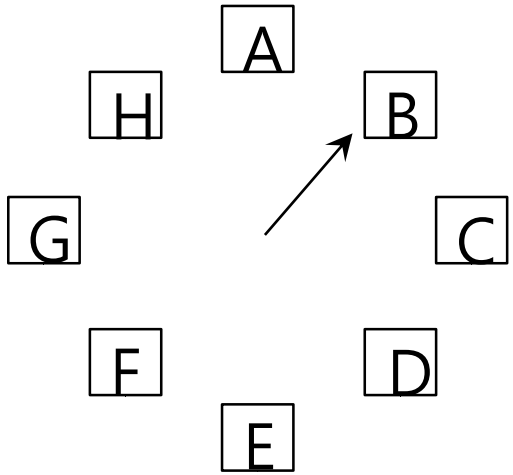


LRU Algorithm

- How to determine “least-recently-used”?
  - need to do some extra accounting work on every memory reference
    - e.g., hardware counters
    - may be expensive
- Approximating LRU with simple hardware support
  - use a **reference bit** in page table entry (PTE)
    - “accessed bit”, “use bit”
  - Whenever a page is referenced, the reference bit is set to 1 by the hardware
    - OS may clear it

Clock Algorithm

- All pages arranges in a circular list
  - A clock hand points to some particular page to begin with; the algorithm continues until it finds a reference bit that is 0 (i.e., not used)



reference bit	action
0	Evict the page
1	Clear **reference bit** and advance hand

Considering Dirty Pages

- Page table entry includes a modified bit (a.k.a dirty bit)
  - Page has been modified (i.e., dirty bit is 1)
    - it must be written back to disk to evict it
  - Page has not been modified (i.e., dirty bit is 0)
    - the eviction is free, i.e., don’t need to write to disk

Agenda

- ~~Recap~~
- ~~Page Replacement Algorithms~~
  - ~~FIFO (First-In-First-Out)~~
  - ~~LRU (Least-Recently-Used)~~
  - ~~Clock~~

Questions?

*acknowledgement: slides include content from “Modern Operating Systems” by A. Tanenbaum, “Operating Systems Concepts” by A. Silberschatz etc., “Operating Systems: Three Easy Pieces” by R. Arpaci-Dusseau etc., and anonymous pictures from internet.