Swapping

Glenn Bruns
CSUMB

Lecture Objectives

After this lecture, you should be able to:

- Explain the mechanics of swapping in an OS
- Manually perform some page replacement policies:
  - Belady’s optimal policy, FIFO, LRU, Random
- Characterize the policies in terms of average memory time and cache hit rates

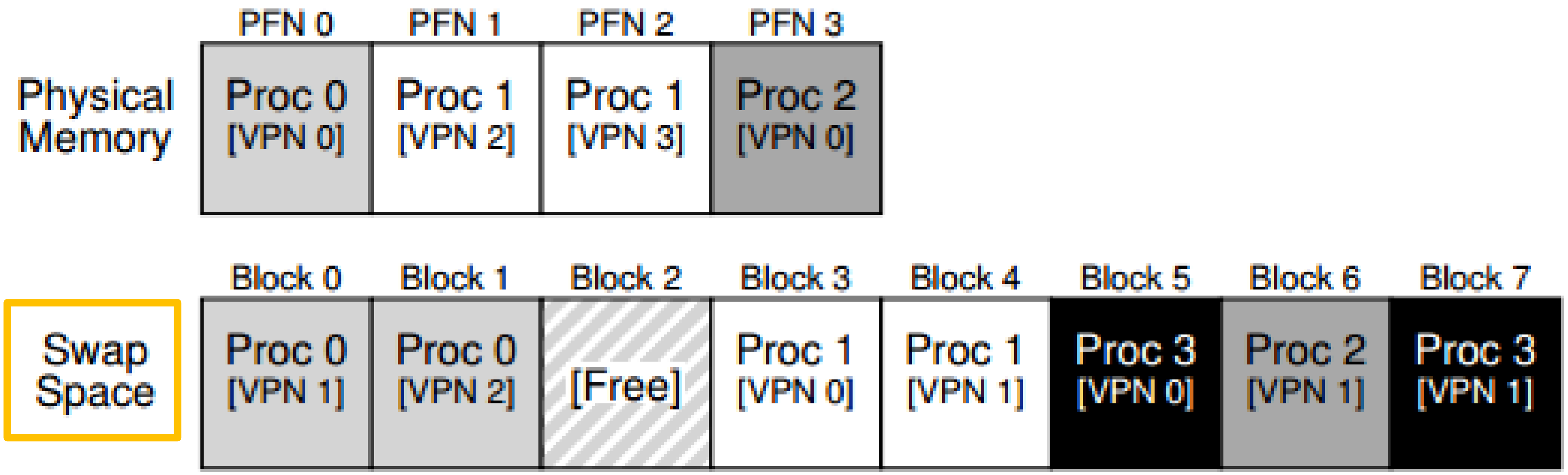
Problem: not enough memory

- OS may be running many processes at once
- Physical memory may not be large enough

Solution ideas

- Reduce size of address space available to processes
- Use disk – let programmers figure out how to move code and data from memory to disk
- Use disk – have OS figure out how to split address space between memory and disk

Pages in memory and on disk

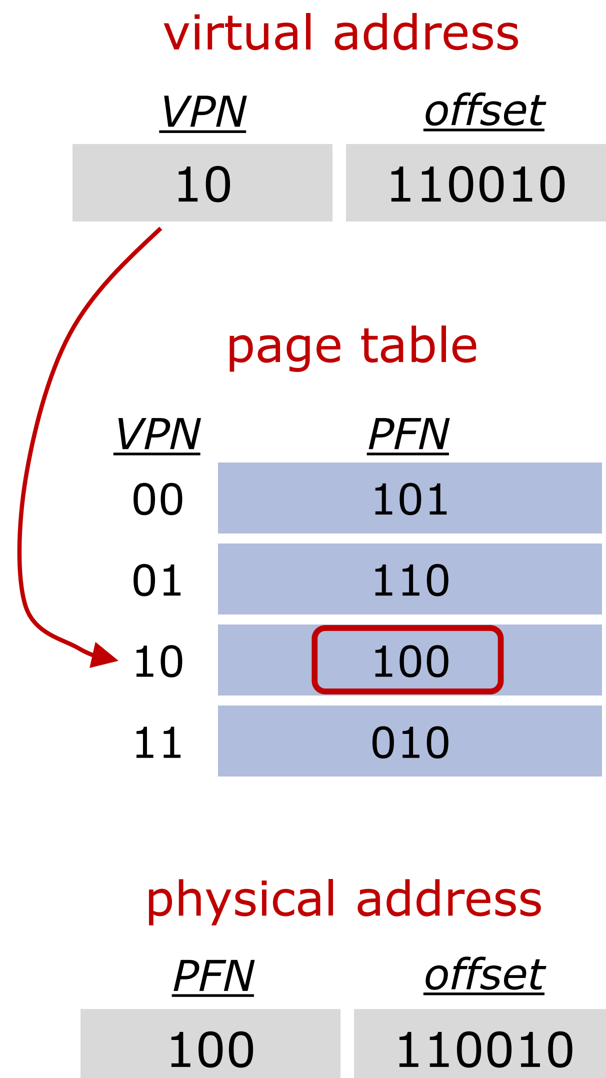


In this tiny example, 4 pages of physical memory, and 8 pages of swap space.

Review: simple paging

Address translation process:

- use VPN as index into page table
- get physical frame number (PFN) from page table entry
- get physical address by combining PFN and offset part of virtual address



Review: simple paging with TLB

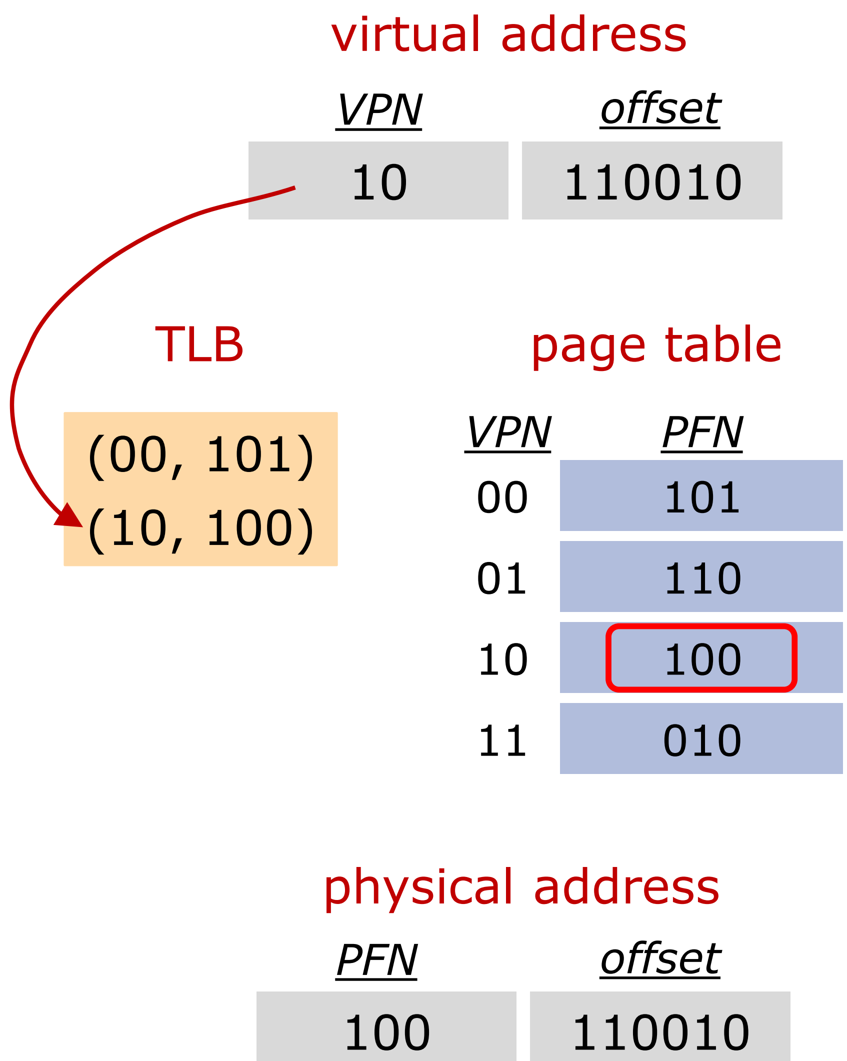
<u>Address translation process</u>:

Lookup PFN in TLB, using VPN as key

if TLB miss:

- use VPN as index into page table
- read PFN from page table entry
- put PFN in TLB
- retry instruction

get physical address by combining PFN and offset part of virtual address



Mechanics of Swapping

- present bit in page table shows whether a page is present in memory
- hardware raises a page fault if a virtual page is not present

page fault handler:

- finds the disk address of the page in the page table
- issues a disk request to fetch the page into memory
- updates page table (once disk I/O completes)
- retries instruction

page table

00	1	12
01	0	120543
10	1	72
11	1	100

present bit

PFN or disk address

NEW!

Adding in swapping

Address translation process:

```
Lookup PFN in TLB, using VPN as key

if TLB miss:
      use VPN as index into page table
      if present bit of page table entry = 0:
          raise page fault
      else:
         read PFN from page table entry
         put PFN in TLB
         retry instruction

get physical address by combining PFN and
offset part of virtual address
```

Swapping: policy

With swapping:

- we can think of memory as a cache for disk

When a page needs to be swapped in, but memory is full, we need to decide which page to swap out.

This is the page replacement policy.

This kind of policy is needed whenever caching is used.

FIFO policy

Evict the "oldest" cache element

3 element cache

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

Least-recently used policy

Evict the least-recently used cache element

3 element cache

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

Random policy

Evict a random cache element

3 element cache

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	2, 0, 1
2	Hit		2, 0, 1
1	Hit		2, 0, 1

Optimal policy (Belady)

Evict cache element to be used furthest in future

3 element cache

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

Why discuss optimal policy?

Optimal policy can't be used in realistic situations

But it serves as a good benchmark.

For example, if we do almost as well as optimal policy, we can't do much better.

Metrics for cache replacement policies

1. Cache hit rate
2. Average memory access time (AMAT)

Suppose we have 1000 memory accesses, and 950 result in cache hits.

What is the Cache hit rate?

Calculate the cache hit rates for the examples on past slides.

Average Memory Access Time

Suppose we have 1000 memory accesses and 950 of them are cache hits.

Also, it takes 100 ns to access memory (the cache), and 10 ms to access disk.

What's the AMAT?

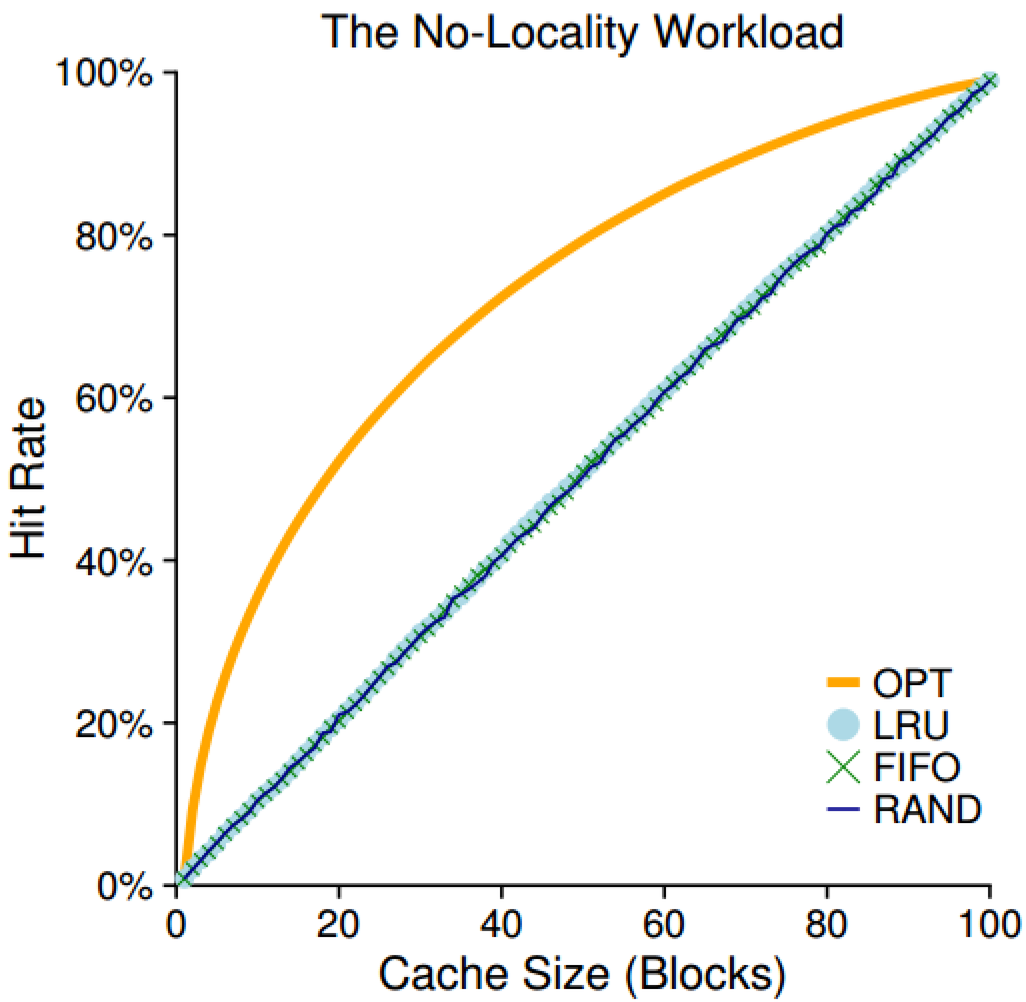
AMAT = (0.95 * 10 ns) + (0.05 * 10 ms)

~ 0.50 ms

What if the cache hit rate is 99%?

~ 0.10 ms

Simulation with no-locality workload

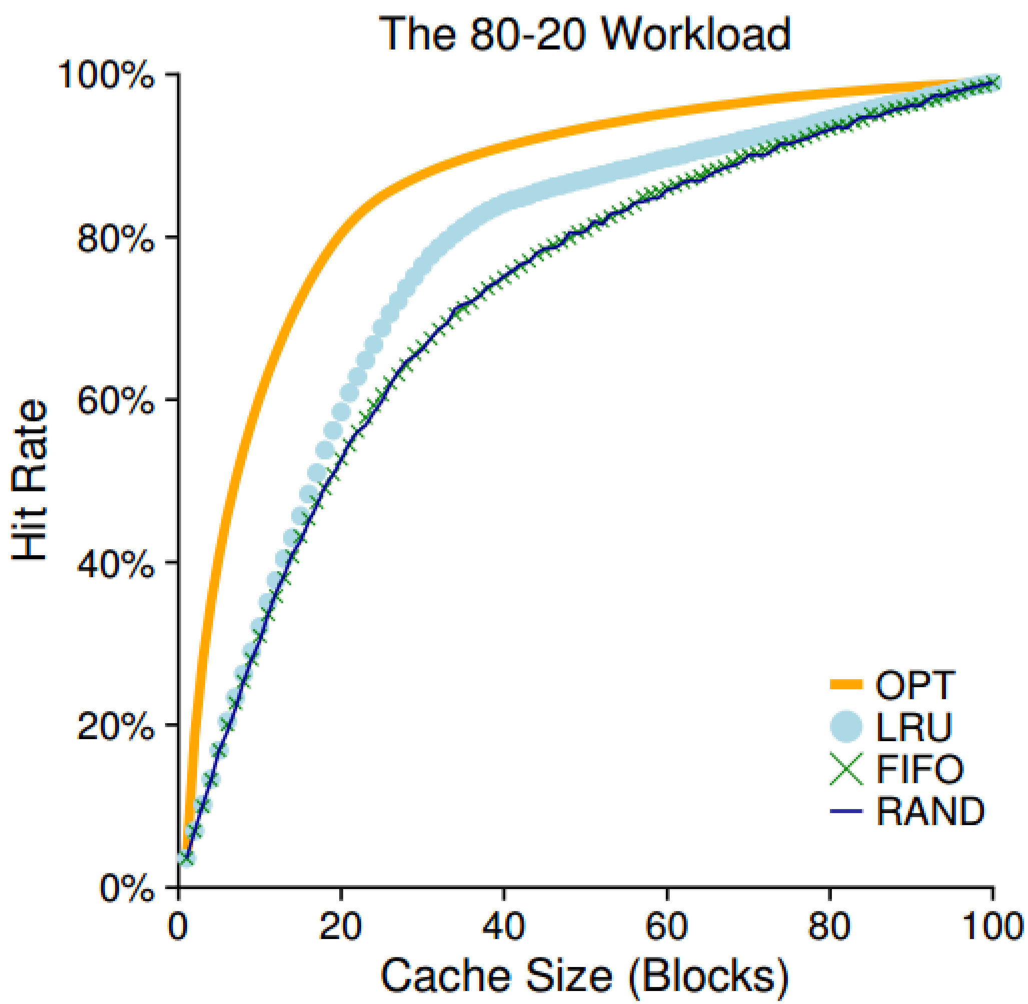


If memory accesses are completely random, all realistic policies have the same hit rate.

Cache size = 40 blocks → hit rate = 40%

(plot from OSTEP)

Simulation with 80:20 workload

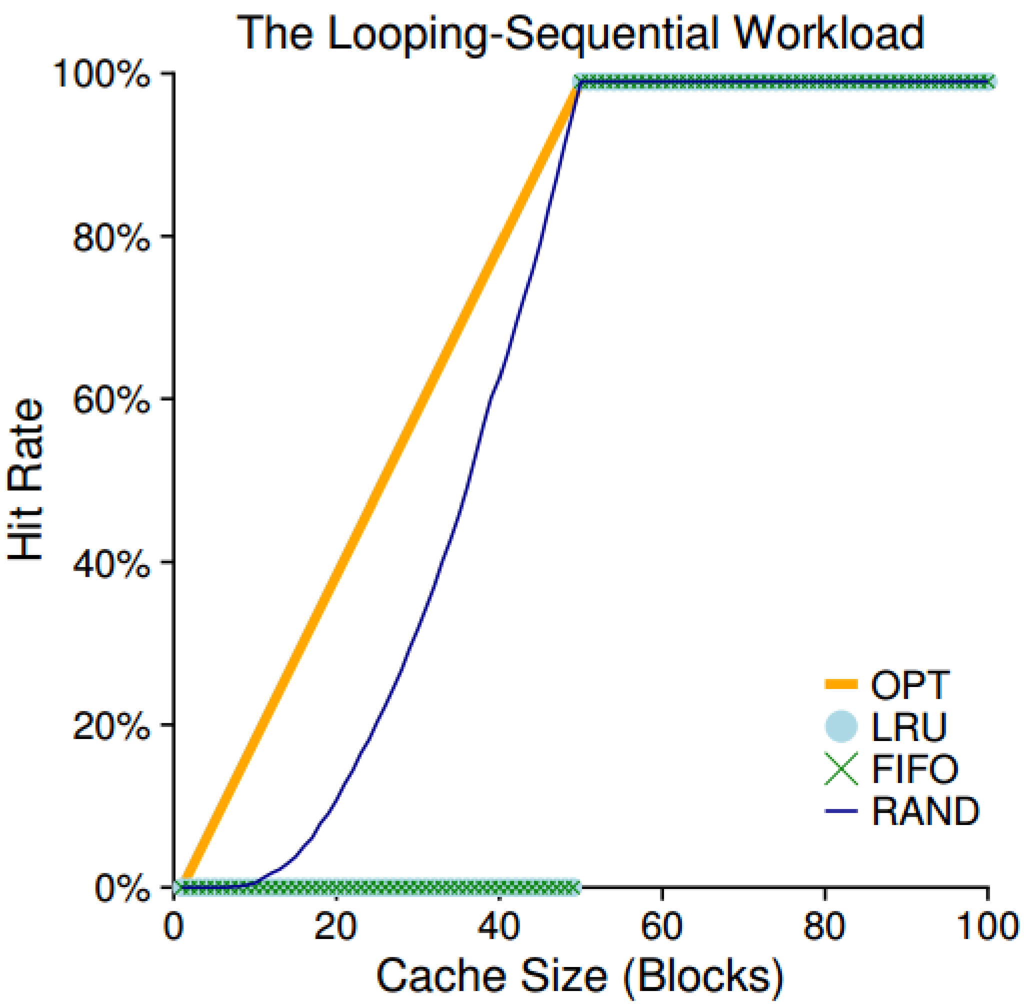


Some locality: 80% of accesses made to 20% of memory.

At cache size = 40 blocks, LRU gives 82% hit rate, optimal gives 90%.

(plot from OSTEP)

Simulation with looping workload



Access pages 0,1,2,…,49 and repeat many times.

Shows strength of random policies.

(plot from OSTEP)

Summary

Swapping is used when insufficient memory for all current processes

Mechanism:

- a page fault is raised if the page table shows the page is not present in memory
- the page fault handler of the OS will then swap the page in from disk

Policy:

- In swapping (like in any situation with caching), we need a page replacement policy
- We looked at these page-replacement policies: FIFO, random, least-recently used (LRU), optimal
- Make sure you can do all these, plus LFU (see text)