IE411: Operating Systems

Swapping and Page Replacement

Swapping

- When main memory fills up, allocating more pages requires some other pages to be evicted from memory
- Evicted pages go to disk (page out)
- swap space (swap file): a reserved space on hard disk for moving pages back and forth

Present bit

- a present bit in the PTE is used to indicate whether a page is present in memory (1) or on disk (0)
- if the present bit is 0, then the other bits in the PTE actually store the disk address of the page
- after paging in a page, the present bit is set to 1 and the content of PTE is updated accordingly

Page fault

- Page faults occur when the CPU sees a present bit set to zero when doing a virtual address translation
- While the OS works to retrieve the page from disk, the process is blocked
- This will take quite a bit of time, so it's best to overlap this request with execution of another processes

Q

If every physical page on a machine with 4GB of RAM is in use and we need to evict a page to make room for another, how many pages do we have to choose from? (assume pages are 4kB)

(A) ∼10k

(B) ∼100k

(C) ∼1M

(D) ∼10M

Q

If every physical page on a machine with 4GB of RAM is in use and we need to evict a page to make room for another, how many pages do we have to choose from? (assume pages are 4kB)

(A) ∼10k

(B) ∼100k

(C) ∼1M

(D) ∼10M

4GB memory $= 2^{32}/2^{12} = 2^{20}$ pages of physical memory!

Page replacement policy

- In the previous example, there are a lot of pages to choose from

Page replacement policy

- In the previous example, there are a lot of pages to choose from
- The choice of which page to evict is called the page replacement policy
- Different policies
  - FIFO
  - Random
  - LRU

Optimal

- The optimal policy is to evict the page that will be needed the furthest in the future
- Provably results in the maximum achievable hit rate

Optimal

- The optimal policy is to evict the page that will be needed the furthest in the future
- Provably results in the maximum achievable hit rate
- Unfortunately it isn't possible to implement
  - as we cannot look into the future!

Example

- Use a cache of size 3
- And accesses of the following pages:
  - 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1

Example

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

Compute the hit rate

Access	Hit/Miss?
0	Miss
1	Miss
2	Miss
0	Hit
1	Hit
3	Miss
0	Hit
3	Hit
1	Hit
2	Miss
1	Hit

- Hits / (Hits + Misses)

Compute the hit rate

Access	Hit/Miss?
0	Miss
1	Miss
2	Miss
0	Hit
1	Hit
3	Miss
0	Hit
3	Hit
1	Hit
2	Miss
1	Hit

- Hits / (Hits + Misses)
  - 6/(6+5) = 54.5%

Compute the hit rate

Access	Hit/Miss?
0	Miss
1	Miss
2	Miss
0	Hit
1	Hit
3	Miss
0	Hit
3	Hit
1	Hit
2	Miss
1	Hit

- Hits / (Hits + Misses)
  - $6/(6+5) = 54.5\%$
- But some of the misses are compulsory!

Compute the hit rate

Access	Hit/Miss?
0	Miss
1	Miss
2	Miss
0	Hit
1	Hit
3	Miss
0	Hit
3	Hit
1	Hit
2	Miss
1	Hit

- Hits / (Hits + Misses)
  - $6/(6+5) = 54.5\%$
- But some of the misses are compulsory!
- Better measure excludes them
  - $6/(6+1) = 85.7\%$

FIFO

- Since we can't implement OPT, let's do something really simple (and $O(1)$)
- Whatever was first into the cache gets evicted

Example

Access	Hit/Miss?	Evict	Resulting Cache State	
0	Miss		First-in→	0
1	Miss		First-in→	0, 1
2	Miss		First-in→	0, 1, 2
0	Hit		First-in→	0, 1, 2
1	Hit		First-in→	0, 1, 2
3	Miss	0	First-in→	1, 2, 3
0	Miss	1	First-in→	2, 3, 0
3	Hit		First-in→	2, 3, 0
1	Miss	2	First-in→	3, 0, 1
2	Miss	3	First-in→	0, 1, 2
1	Hit		First-in→	0, 1, 2

Quiz

Access	Hit/Miss?
0	Miss
1	Miss
2	Miss
0	Hit
1	Hit
3	Miss
0	Miss
3	Hit
1	Miss
2	Miss
1	Hit

What is the hit rate (excluding compulsory misses) of FIFO in this example?

(A) 4/11

(B) 4/7

(C) 11/4

(D) 7/11

What about implementation?

- In FIFO items are evicted in the order they are inserted
- Key idea: use a queue of page numbers
  - We replace the page at the head of the queue
  - When a page is brought into memory, it is inserted at the tail of the queue

Belady's Anomaly

- In general you would expect cache hit rate to increase when cache gets larger
- Does this always happen?

Belady's Anomaly

- In general you would expect cache hit rate to increase when cache gets larger
- Does this always happen?
- Not necessarily for FIFO!
  - You can sometimes get a worse hit rate with a larger cache size

Example

- Accesses: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

(a) size 3

Access	Hit	State (after)
1	no	1
2	no	1,2
3	no	1,2,3
4	no	2,3,4
1	no	3,4,1
2	no	4,1,2
5	no	1,2,5
1	yes	1,2,5
2	yes	1,2,5
3	no	2,5,3
4	no	5,3,4
5	yes	5,3,4

(b) size 4

Access	Hit	State (after)
1		
2		
3		
4		
1		
2		
5		
1		
2		
3		
4		
5		

Example

- Accesses: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

(a) size 3

Access	Hit	State (after)
1	no	1
2	no	1,2
3	no	1,2,3
4	no	2,3,4
1	no	3,4,1
2	no	4,1,2
5	no	1,2,5
1	yes	1,2,5
2	yes	1,2,5
3	no	2,5,3
4	no	5,3,4
5	yes	5,3,4

(b) size 4

Access	Hit	State (after)
1	no	1
2	no	1,2
3	no	1,2,3
4	no	1,2,3,4
1	yes	1,2,3,4
2	yes	1,2,3,4
5	no	2,3,4,5
1	no	3,4,5,1
2	no	4,5,1,2
3	no	5,1,2,3
4	no	1,2,3,4
5	no	2,3,4,5

Random

- Oh, when things get hard, just roll the dice!
- Simply pick a random victim to evict, not even trying to be intelligent

Least Recently Used (LRU)

- Replace page that hasn't been used for the longest time

Least Recently Used (LRU)

- Replace page that hasn't been used for the longest time
- Programs have locality, so if something not used for a while, unlikely to be used in the near future.
- Seems like LRU should be a good approximation to OPT.

Least-Recently-Used (LRU)

- Accesses: 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1

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

Least-Recently-Used (LRU)

- Accesses: 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1

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

Least-Recently-Used (LRU)

- Accesses: 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1

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

Least-Recently-Used (LRU)

- Accesses: 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1

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

Least-Recently-Used (LRU)

- Accesses: 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1

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

Least-Recently-Used (LRU)

- Accesses: 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1

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

Least-Recently-Used (LRU)

- Accesses: 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1

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

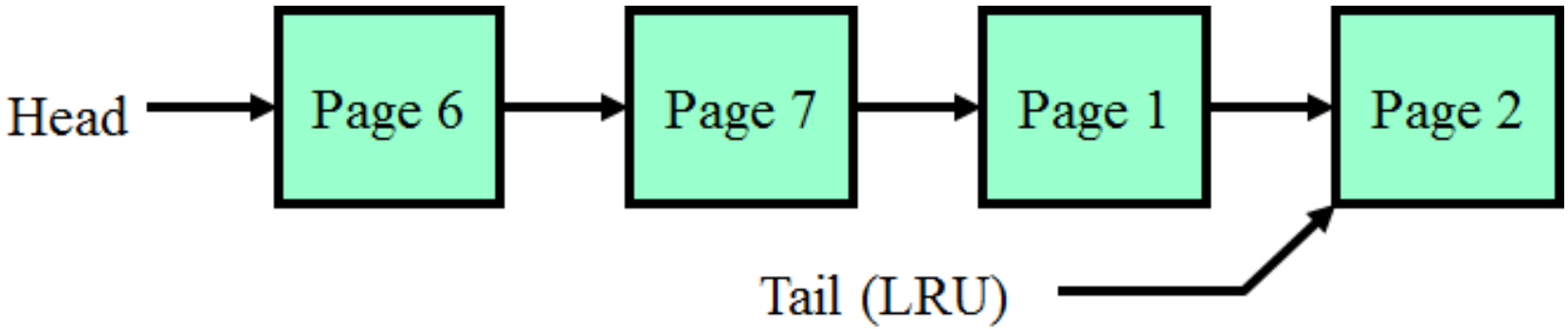
Least-Recently-Used (LRU)

- Accesses: 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1

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

What about implementation?

- We can use a list!



- On each use, remove page from list and place at head
- LRU page is at tail
- List must be updated at every memory reference; in FIFO we only have to update the list when evicting

OPT vs. LRU

- Suppose we have 3 page frames, 4 virtual pages (0–3), and following reference stream:
  - 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1
- Consider OPT page replacement:

OPT vs. LRU

- Suppose we have 3 page frames, 4 virtual pages (0–3), and following reference stream:
  - 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1
- Consider OPT page replacement:

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

OPT vs. LRU

- Suppose we have 3 page frames, 4 virtual pages (0–3), and following reference stream:
  - 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1
- Consider OPT page replacement:

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

- Here LRU would work equally well as OPT

OPT vs. LRU

- Suppose we have 3 page frames, 4 virtual pages (0–3), and following reference stream:
  - 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1
- Consider OPT page replacement:

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

- Here LRU would work equally well as OPT
  - in general, won't be true!

LRU vs. FIFO

- As before suppose we have 3 page frames, 4 virtual pages (0–3)
- Consider the following reference stream:
  - 0, 1, 2, 3, 0, 1, 2, 3
- Here LRU performs quite badly (same as FIFO)
  - 8 misses

LRU vs. FIFO

- As before suppose we have 3 page frames, 4 virtual pages (0–3)
- Consider the following reference stream:
  - 0, 1, 2, 3, 0, 1, 2, 3
- Here LRU performs quite badly (same as FIFO)
  - 8 misses
  - when referencing 3, evicting 0 is a bad choice, since need 0 right away

LRU vs. FIFO

- As before suppose we have 3 page frames, 4 virtual pages (0–3)
- Consider the following reference stream:
  - 0, 1, 2, 3, 0, 1, 2, 3
- Here LRU performs quite badly (same as FIFO)
  - 8 misses
  - when referencing 3, evicting 0 is a bad choice, since need 0 right away
- OPT does much better!
  - 5 misses

LRU vs. FIFO

- As before suppose we have 3 page frames, 4 virtual pages (0–3)
- Consider the following reference stream:
  - 0, 1, 2, 3, 0, 1, 2, 3
- Here LRU performs quite badly (same as FIFO)
  - 8 misses
  - when referencing 3, evicting 0 is a bad choice, since need 0 right away
- OPT does much better!
  - 5 misses
  - when referencing 3, evict 2 since it is referenced farthest in future