Operating Systems Z02-54c

Page Replacement Algorithms

FIFO, LRU, OPT ,ALRU, RAND

Mustafa Tayyip BAYRAM 257639

OVERVIEW

A computer system has a limited amount of memory. Adding more memory physically is very costly. Therefore most modern computers use a combination of both hardware and software to allow the computer to address more memory than the amount physically present on the system. This extra memory is actually called **Virtual Memory.**

Virtual Memory is a storage allocation scheme used by the Memory Management Unit(MMU) to compensate for the shortage of physical memory by transferring data from RAM to disk storage. It addresses secondary memory as though it is a part of the main memory. Virtual Memory makes the memory appear larger than actually present which helps in the execution of programs that are larger than the physical memory.

Virtual Memory can be implemented using Paging method.

➢ Paging :

In an operating system that uses paging for memory management, a page replacement algorithm is needed to decide which page needs to be replaced when new page comes in.

Page Fault – A page fault happens when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory.

Since actual physical memory is much smaller than virtual memory, page faults happen. In case of page fault, Operating System might have to replace one of the existing pages with the newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce the number of page faults.

Page Replacement Algorithms

1. **First In First Out (FIFO) :** This is the simplest page replacement algorithm. In this algorithm, the OS maintains a queue that keeps track of all the pages in memory, with the oldest page at the front and the most recent page at the back.
   When there is a need for page replacement, the FIFO algorithm, swaps out the page at the front of the queue, that is the page which has been in the memory for the longest time.

- Advantages :
  - Simple and easy to implement.
  - Low overhead.
- Disadvantages :
  - Poor performance.
  - Does not consider the frequency of use or last used time, simply replaces the oldes page.
- For Example :
  - Consider the page reference string of size 12: 1, 2, 3, 4, 5, 1, 3, 1, 6, 3, 2, 3 with frame size 4(i.e. maximum 4 pages in a frame).

1	2	3	4	5	1	3	1	6	3	2	3

1	1	1	1	5	5	5	5	5	5	2	2
	2	2	2	2	1	1	1	1	1	1	1
		3	3	3	3	3	3	6	6	6	6
			4	4	4	4	4	4	3	3	3
M	M	M	M	M	M	H	H	M	M	M	H

M = Miss
H = Hit

  - Total Page Fault = 9

  - Initially, all 4 slots are empty, so when 1, 2, 3, 4 came they are allocated to the empty slots in order of their arrival. This is page fault as 1, 2, 3, 4 are not available in memory.

  - When 5 comes, it is not available in memory so page fault occurs and it replaces the oldest page in memory, i.e., 1.
  - When 1 comes, it is not available in memory so page fault occurs and it replaces the oldest page in memory, i.e., 2.
  - When 3,1 comes, it is available in the memory, i.e., Page Hit, so no replacement occurs.
  - When 6 comes, it is not available in memory so page fault occurs and it replaces the oldest page in memory, i.e., 3.
  - When 3 comes, it is not available in memory so page fault occurs and it replaces the oldest page in memory, i.e., 4.

- When 2 comes, it is not available in memory so page fault occurs and it replaces the oldest page in memory, i.e., 5.
- When 3 comes, it is available in the memory, i.e., Page Hit, so no replacement occurs.
- **Page Fault ratio = 9/12 i.e. total miss/total possible cases**

2. **Least Recently Used (LRU) :** Least Recently Used page replacement algorithm keeps track of page usage over a short period of time. It works on the idea that the pages that have been most heavily used in the past are most likely to be used heavily in the future too. In LRU, whenever page replacement happens, the page which has not been used for the longest amount of time is replaced.
   - Advantages :
     - Efficient
   - Disadvantages :
     - Complex Implementation.
     - Expensive.
     - Requires hardware support.
   - For example :

1	2	3	4	5	1	3	1	6	3	2	3

1	1	1	1	5	5	5	5	5	5	2	2
	2	2	2	2	1	1	1	1	1	1	1
		3	3	3	3	3	3	3	3	3	3
			4	4	4	4	4	6	6	6	6
M	M	M	M	M	M	H	H	M	H	M	H

M = Miss
H = Hit

   - Total Page Fault = 8
   - Initially, all 4 slots are empty, so when 1, 2, 3, 4 came they are allocated to the empty slots in order of their arrival. This is page fault as 1, 2, 3, 4 are not available in memory.
   - When 5 comes, it is not available in memory so page fault occurs and it replaces 1 which is the least recently used page.
   - When 1 comes, it is not available in memory so page fault occurs and it replaces 2.
   - When 3,1 comes, it is available in the memory, i.e., Page Hit, so no replacement occurs.
   - When 6 comes, it is not available in memory so page fault occurs and it replaces 4.
   - When 3 comes, it is available in the memory, i.e., Page Hit, so no replacement occurs.
   - When 2 comes, it is not available in memory so page fault occurs and it replaces 5.

- When 3 comes, it is available in the memory, i.e., Page Hit, so no replacement occurs.
- **Page Fault ratio = 8/12**

3. **Optimal Page Replacement (OPT) :** Optimal Page Replacement algorithm is the best page replacement algorithm as it gives the least number of page faults. It is also known as OPT.

In this algorithm, pages are replaced which would not be used for the longest duration of time in the future, i.e., the pages in the memory which are going to be referred farthest in the future are replaced.

This algorithm was introduced long back and is difficult to implement because it requires future knowledge of the program behavior. However, it is possible to implement optimal page replacement on the second run by using the page reference information collected on the first run.

- Advantages :
  - Easy to Implement
  - Simple data structures are used.
  - Highly efficient.
- Disadvantages :
  - Requires future knowledge of the program.
  - Time-consuming.
- For example :

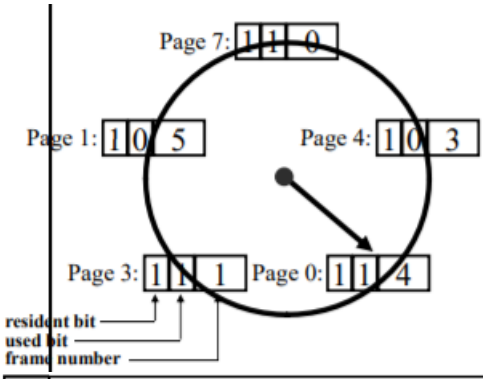
1	2	3	4	5	1	3	1	6	3	2	3

1	1	1	1	1	1	1	1	6	6	6	6
	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3	3	3	3
			4	5	5	5	5	5	5	5	5
M	M	M	M	M	H	H	H	M	H	H	H

M = Miss
H = Hit

  - Total Page Fault = 6
  - Initially, all 4 slots are empty, so when 1, 2, 3, 4 came they are allocated to the empty slots in order of their arrival. This is page fault as 1, 2, 3, 4 are not available in memory.
  - When 5 comes, it is not available in memory, so page fault occurs and it replaces 4 which is going to be used farthest in the future among 1, 2, 3, 4.
  - When 1,3,1 comes, they are available in the memory, i.e., Page Hit, so no replacement occurs.
  - When 6 comes, it is not available in memory, so page fault occurs and it replaces 1.
  - When 3, 2, 3 comes, it is available in the memory, i.e., Page Hit, so no replacement occurs.
  - **Page Fault ratio = 6/12**

4. **Approximate Least Recently Used (ALRU) :** Maintains a circular list of pages resident in memory and uses a clock (or used/referenced) bit to track how often a page is accessed. The bit is set whenever a page is referenced. Clock hand sweeps over pages looking for one with used bit = 0. Replace pages that have not been referenced for one complete revolution of the clock.



➢

Clock Page Replacement
Example

Time	0	1	2	3	4	5	6	7	8	9	10
Requests		*c*	*a*	*d*	*b*	*e*	*b*	*a*	*b*	*c*	*d*
Page Frames 0	*a*	*a*	*a*	*a*	*a*						
Page Frames 1	*b*	*b*	*b*	*b*	*b*						
Page Frames 2	*c*	*c*	*c*	*c*	*c*						
Page Frames 3	*d*	*d*	*d*	*d*	*d*						
Faults											

Page table entries for resident pages:

1	*a*
1	*b*
1	*c*
1	*d*

➢

Clock Page Replacement
Example

Time	0	1	2	3	4	5	6	7	8	9	10
Requests		*c*	*a*	*d*	*b*	*e*	*b*	*a*	*b*	*c*	*d*
Page Frames 0	*a*	*a*	*a*	*a*	*a*	***e***	*e*	*e*	*e*	*e*	***d***
Page Frames 1	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*	*b*
Page Frames 2	*c*	*c*	*c*	*c*	*c*	*c*	*c*	***a***	*a*	*a*	*a*
Page Frames 3	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*	***c***	*c*
Faults						•		•		•	•

Page table entries for resident pages:

1	*a*	1	*e*	1	*e*	1	*e*	1	*e*	1	*e*	1	*e*	1	*d*
1	*b*	0	*b*	1	*b*	0	*b*	1	*b*	1	*b*	1	*b*	0	*b*
1	*c*	0	*c*	0	*c*	1	*a*	1	*a*	1	*a*	0	*a*
1	*d*	0	*d*	0	*d*	0	*d*	0	*d*	1	*c*	0	*c*

➢

5. RAND:

- In page replacement using a random algorithm (random) every time a page fault occurs, the replaced page is randomly selected. This technique does not use any information in determining which pages to replace, all pages in main memory have the same weight to be selected. How to select random or arbitrary pages, including the page that is currently referenced (pages that should not be replaced). This random technique is very bad, the experiment shows the random algorithm causes a very high frequency of page faults.

TABLE 1. Case Study 1

Page queue	0	1	3	4		2	0	0	4		1	3	3	3
Random Replacement	0	0	0*	4		4*	0*	^	4		4	4*	3*	^
		1	1	1*		2	2	^	2		2	2	2	^
			3	3		3	3	^	3*		1*	3	3	^
Page Fault	x	x	x	x		x	x	^	x		x	x	x	^

x = miss

- ^ = hit