National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Electrical Engineering

Faculty Member: Dr. Nasir Mahmood Dated: 4/5/2018

Course/Section: BSCS-6-A Semester: 4th

GROUP # 02

Computer Organization and

Achitecture (EE321)

Lab #9 Page Replacement Algorithms

Name	Reg. no.	Report Marks / 5	Viva Marks / 10	Total/15
Arifullah Jan	18943			
Bilal Khalid	128608			
Waqas Yaseen	196819			

Objectives

At the end of this lab you should be able understand Page Replacement Algorithms and their comparison in term of performance

Lab 10

Lab Instructions

- ✓ This lab activity comprises two parts, namely Lab tasks, and Post-Lab Viva session.
- ✓ Each group to upload completed lab report on LMS for grading.
- ✓ The students will start lab task and demonstrate each lab task separately for step-wise evaluation
- ✓ There are related questions in this activity give complete answers. Also provide complete code and results.

Page Replacement Algorithms

First-In First-Out (FIFO)

FIFO is a very simple replacement policy, and is also known as a round robin. The way it works is once the cache is full it simply replaces the first page that was placed in the cache with the desired page, and the next replacement will be the second page placed in the cache and so on….

Random

Random is just that, when the cache is full it randomly picks which page will be replaced.

Optimal

This is the ideal case, and is impossible to perform in practice. This is where the system can look at all the references that will happen in the future. The page that is then chosen to be replaced is the one that is either not being referenced again, or is being referenced again the farthest into the future compared to the other in the cache.

Least Frequently Used (LFU)

LFU keeps a list of all the pages referenced in the cache and how many times they have been referenced in the past. Once the cache becomes full it replaces the page that has been referenced the fewest, with the new one.

Least Recently Used (LRU)

An ideal LRU policy keeps track of how long ago each page was referenced. When a page needs to be replaced in order to make room for a new one, the page that was least recently used is replaced. In other words the page in the cache that hasn't been used for the longest time is replaced. On every reference all of the pages' history counters are incremented by one, and when a hit occurs the history counter for that page is set back to zero.

Least Recently Used K (LRU-K)

LRU-K is a practical implementation of the LRU policy. The basic idea behind the LRU-K replacement policy is that it keeps track of the last K references. When the cache becomes full it replaces the page with the largest K length. However, there can be several pages with the maximum length of K, and when this happens several different routes can be taken to decide which one to

choose. In this tool it cycles through the pages to attempt to keep the same page from being replaced every time it comes across this situation.

LRU-K does not have a higher hit ratio then the ideal LFU policy in most cases. However, LRU-K is more practical because it limits the number of bits used to monitor how recently a value was last used. One example when it does outperform the other policies is the following:

e.g. 1 2 3 4 5 6 8 4 1 2 13 10 15 2 7 9 5 1 2 3 4 5 12 8 6 4 1 2 3 4 5 12 4 8 7 4 6

Number of pages = 10 LRU-K = 4

Adaptive Replacement Cache (ARC)

Recently developed in response to the downfalls of LRU policies. LRU captures only recency and not frequency, and can be easily "polluted" by a scan. A scan is a sequence of one-time use only page requests, and leads to lower performance. ARC overcomes these two downfalls by using four doubly linked lists. Lists T1 and T2 are what is actually in the cache at any given time, while B1 and B2 act as a second level. B1 and B2 contain the pages that have been thrown out of T1 and T2 respectively. The total number of pages therefore needed to implement these lists is 2*C, where C is the number of pages in the cache. T2 and B2 contain only pages that have been used more than once. The lists both use LRU replacement, in which the page removed from T2 is placed into B2. T1 and B1 work similarly together, except where there is a hit in T1 or B1 the page is moved to . The part that makes this policy very adaptive is the sizes of the lists change. List T1 size and T2 size always add up to the total number of pages in the cache. However, if there is a hit in B1, also known as a Phantom Hit (i.e. not real hit in cache), it increases the size of T1 by one and decreases the size of T2 by one. In the other direction, a hit in B2 (Phantom Hit) will increase the size of T2 by one and decrease the size of T1 by one. This allows the cache to adapt to have either more recency or more frequency depending on the workload.

The following sequence shows how ARC out performs all of the other policies.

e.g. 0 1 2 3 0 4 1 2 3 0 1 2 3 0 1 2 4 3

Number of pages = 3

The sequence below demonstrates that ARC doesn't always have the best performance however in many cases the number of cold start misses is reduced.

e.g. 1 2 5 4 3 2 1 5 4 6 8 4 5 2 7 10 1 2 15 15 2 4 6 3 2 21 8 78 6 1 2 3 4 5 6 7 8 9 1 3 22 5 4 9 15 12 9

number of pages = 10

Clock with Adaptive Replacement (CAR)

CAR works in much the same way that ARC does. However, CAR avoids the last two downfalls of LRU that ARC does not. These downfalls are: 1) the overhead of moving a page to the most

recently used position on every page hit, and 2) serialized, in the fact that when moving these pages to the most recently used position the cache is locked so that it doesn't change during this transition. This then causes delays in a multi-thread environment. In reality CAR only overcomes the first downfall, the second still has a presence. CAR uses two clocks instead of lists, a clock is a circular buffer. The two circular buffers are similar in nature to T1 and T2, and it does contain a B1 and B2 as well. The main difference is that each page in T1 and T2 contain a reference bit. When a hit occurs this reference bit is set on. T1 and T2 still vary in size the same way they do in ARC (i.e. Phantom Hits cause these changes). When the cache is full the corresponding clock begins reading around itself (i.e. the buffer) and replaces the first page whose reference bit is zero. It also sets the reference bits back to zero if they are currently set to one. The clock will continue to travel around until it finds a page with reference bit equal to zero to replace.

The CAR policy reduces overhead compared to LRU. This cannot be seen when viewing the simulator, however the CAR policy uses a circular buffer and this removes the overhead of having to move the most-recently used position on every page hit. The following sequence however shows that it can outperform the policies used today (ARC, LRU, LRU-K). CAR does not outperform the ideal LFU in terms of hit ratio, however LFU has much more overhead because it keeps track of the number of times a value is used for an infinite number of values.

e.g. 1 2 3 4 5 6 1 2 5 4 8 9 6 2 5 4 10 2 5 4

number of pages = 4 LRU-K= 3

LAB TASK 1

Online Simulator: http://www.ecs.umass.edu/ece/koren/architecture/PReplace/

1) Provide a sequence of 15 page references that can show the following. Provide the page frame number(s) that satisfy that condition.

1. LRU is better (greater hit ratio) than FIFO
   Page references: 1 2 5 7 2 5 1 5 1 4 1 5 2 1 4
   Number of Page Frames: 3
   Hit Ratio: LRU: 0.467 FIFO: 0.333

2. LRU and FIFO are equal
   Page references: 1 2 5 7 2 5 1 5 1 4 1 5 2 1 4
   Number of Page Frames: 2
   Hit Ratio: LRU: 0.2 FIFO: 0.2

3. LRU is performs worse than FIFO
   Page references: 1 2 2 7 2 1 7 1 7 10 9 3 2 5 1
   Number of Page Frames: 2
   Hit Ratio: LRU: 0.267 FIFO: 0.333

4. OPTIMAL has same hit ratio as LRU

Page references: 1 2 2 7 2 1 7 1 7 10 9 3 2 5 1
Number of Page Frames: 8
Hit Ratio: LRU: 0.533 OPTIMAL: 0.533

5. All the three have the same hit ratio.
   Page references: 1 2 2 7 2 1 7 1 7 10 9 3 2 5 1
   Number of Page Frames: 7
   Hit Ratio: LRU: 0.533 FIFO: 0.533 OPTIMAL: 0.533

2) Define page table and translation lookaside buffer (TLB). Give an analogy in the memory hierarchy to this page table-TLB pair.

Translation lookaside buffer: It is a memory cache that contains virtual address and its corresponding physical address. It is attached with Memory management unit (MMU) and can be either logical or physical.

Page Table: It is a data structure that similarly stores virtual address mapped to physical addresses data. This data structure is managed by OS and is stored in memory.

Analogy: This page table – TLB can be mapped to memory hierarchy in a sense like: Finding a particular block in a cache from a physical address of a memory block.

LAB TASK 2

Online Simulator: http://www.ecs.umass.edu/ece/koren/architecture/RepPolicies/

1) Use the following input sequence. 1 1 1 2 2 3 4 1 4 5 6 9 8 0 1 2 3 5 6 23 6 6 6 7. Use k=2, page frames as 3.

List the hit rates in all the different replacement algorithms and find which algorithms form the lower and upper bounds.

How different is LRU-k from LRU. How does the value of k affect the LRU-k result? Is there an optimal number?

2) Use the following input sequence. 0 1 2 3 4 5 6 7 8 16 17 9 10 11 12 28 29 30 8 9 10 4 5 12 4 5.Find the page frame numbers for each of the following. List the optimal values for each case.

(a) FIFO performs better than LRU.
(b) LRU, FIFO, LFU and CAR all perform equally.
(c) LFU performs better than LRU, FIFO, CAR and ARC
(d) All except Random are equal

3) Page Replacement Algorithms

(a) What is the main goal of all the techniques suggested?

Main goal is to devise a better algorithm that can have maximum hit ratio for accessing block from cache. Greater the hit ratio more is the speed of processing as data is to be fetched from cache which is faster than the memory.

(b) What factors does the replacement policy depend upon for its effectiveness?

1. Material of the cache used.
2. Cache Size
3. Total hits and miss ratios.
4. Page Size.
5. Whether read or write is done on the cache
6. Read and Write speed.