John Gangemi
Raj Patel

Project 3: Measuring the Performance of Page Replacement Algorithm on Real Traces

Introduction

This lab dealt with page replacement policies and choosing the best policy to make sure you get the least amount of memory misses and consequently increasing the amount of memory hits.Page replacement policies are used to decide which pages to keep in memory and which to evict to yield the best hit rate .

The purpose of the lab is to write a memory simulator and evaluate memory performance using the different traces (which is a recording of a running program taken from the SPEC benchmark).We were specifically tasked with using the following six paging policies: Random(rand),Least Recently Used(LRU),Least Frequently Used(LFU),Most Frequently Used (MFU),OPT(Optimal Page replacement) and FIFO (First-in-First-Out).

Below is a brief description of each of these page.

Rand: This policy simply picks a random page in memory and evicts it and replaces it with the current page that you are trying to place in memory.

LRU: Replaces the least recently used pages in the memory with the current page you are trying to place in memory.

LFU: Replaces the least frequently used page in memory when a new page from the the page table needs to be added to memory.

MFU: The opposite of LFU. Replaces the most frequently used pages in memory with the current page you are trying to place in memory.

OPT: Replaces the page that will be accessed furthest in the future from the current page that we need to place into memory. It does this by going through the pages a;ready in memory and seeing where that page is in the page table in relation to the current page that you want to place in memory.

FIFO: Follows the queue structure. The pages are put into a queue where whatever was first to enter the queue is the first one to be replaced (also known as the one on the tail of the queue)

Methods

To start off some calculations need to be performed. It is observed that a 8 hexadecimal address is given in the bzip.trace file which translates to 32 bits. In order to figure out the size of the page table given a virtual address the page size needs to be known which in this case is exactly 4KB

(2^{12}) .Thus the size of the page table is the virtual address divided by the page size as shown by the following equation $2^{32}/ 2^{12} = 2^{20}$. From this result it becomes apparent that the number of bits to correctly identify a page table entry is 20 bits. These 20 bits are from the first 32 bits of the HEX address where each value of the HEX address is 4 bits therefore the first five HEX values define the virtual page number.

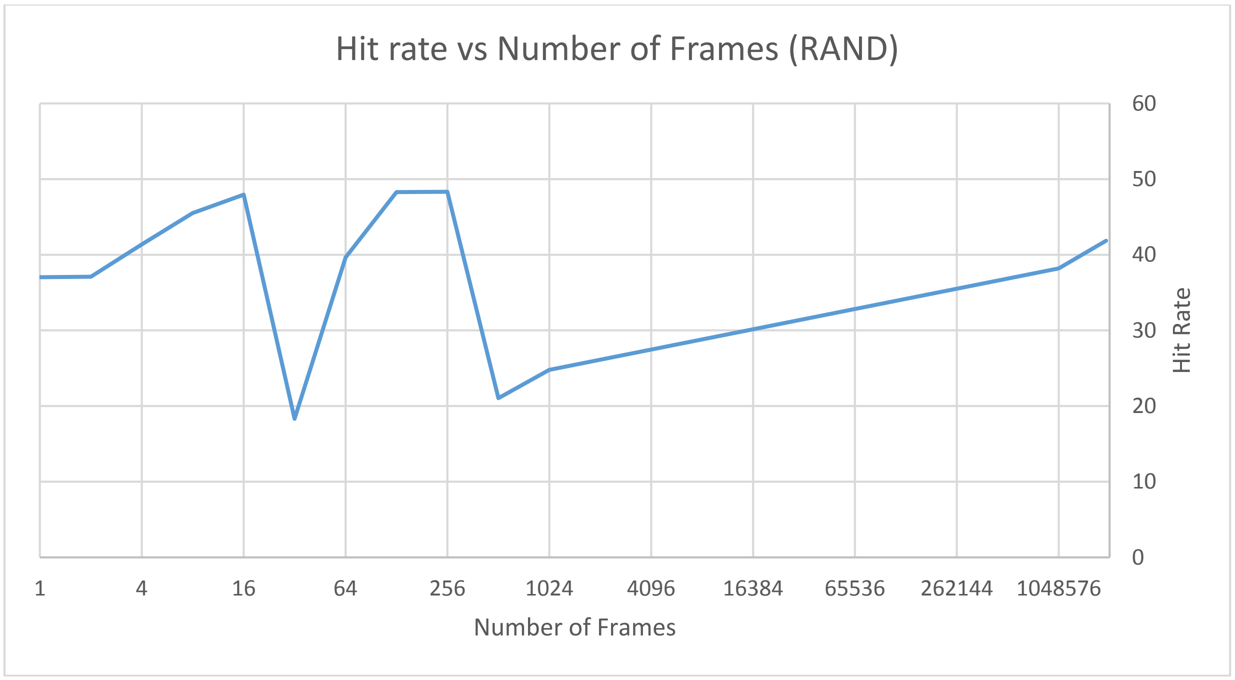
Per guideline of the rubric we decided to use one of the four trace files (bzip.trace specifically) to test the 5 different implementation (opt not included) of our page table replacement algorithms. Each time the algorithm was run the frame size was incremented by a power of 2 upto 2,000,000 (ie 1,2,4,8,16..1024). This allowed us to check the cases where the size of the memory was less than, equal to and greater than the size of the page table.

Results

Below are the results for the different algorithms using the bzip.trace and a brief description explaining the graph and the data.

RAND

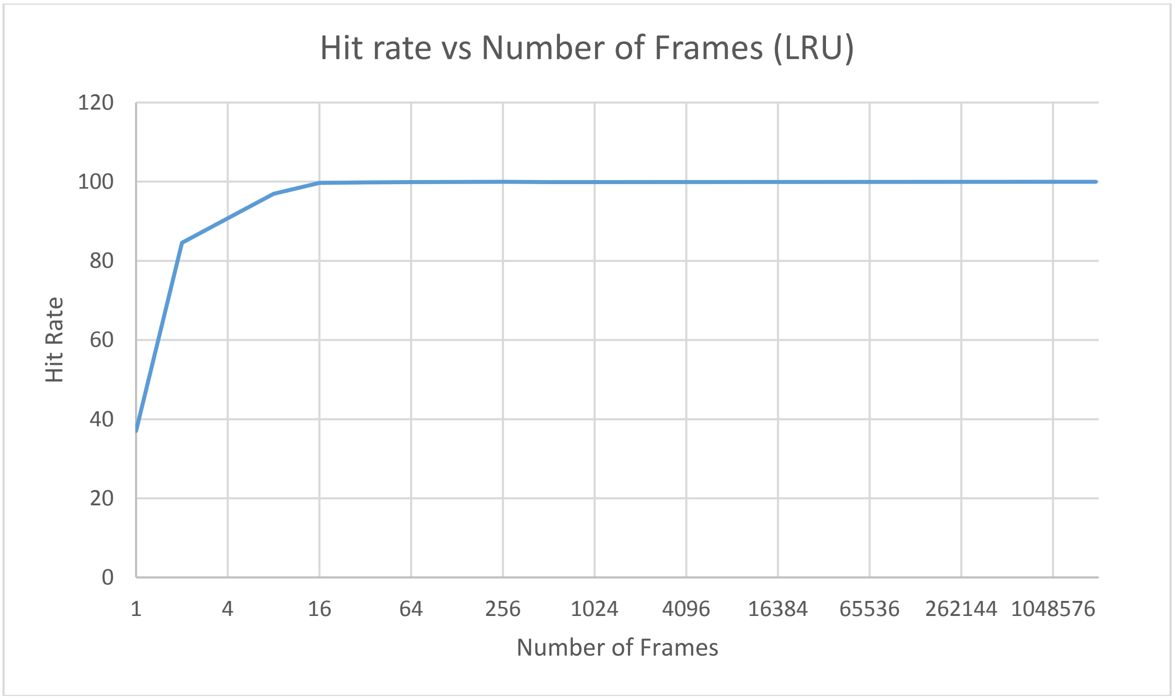
Number of frames	Hit rate
1	37.026
2	37.103
4	41.359
8	45.511
16	47.944
32	18.324
64	39.652
128	48.273
256	48.33
512	21.05
1024	24.792
1048576	38.196
2000000	41.859



Rand overall performed very poorly. This was the case for when the number of frames was greater than, less than and equal to the page size. The highest hit rate was 48 percent which was when the number of frames was 128.

LRU

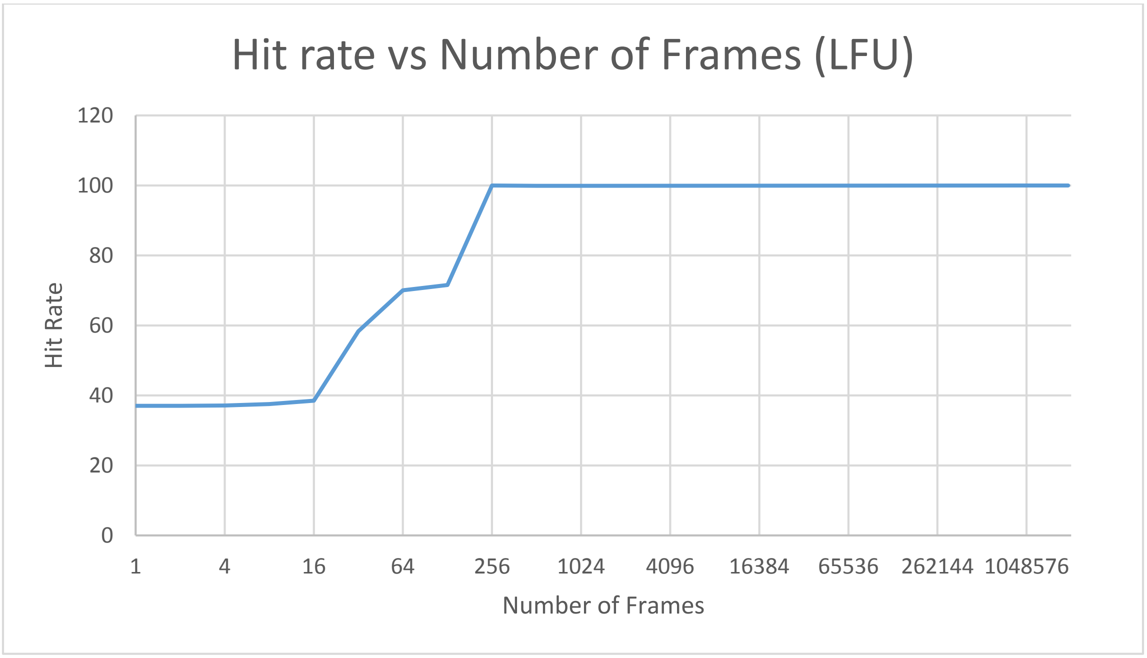
Number of frames	Hit rate
1	37.026
2	84.557
4	90.723
8	96.931
16	99.666
32	99.787
64	99.874
128	99.923
256	99.96
512	99.881
1024	99.881
1048576	99.968
2000000	99.968



LRU was efficient very early on. After the number of frames passed 4 frames the hit rate was 90 percent or above. The best hit rate occurred when the number of frames was 256 and the worst was when there was one frame.

LFU

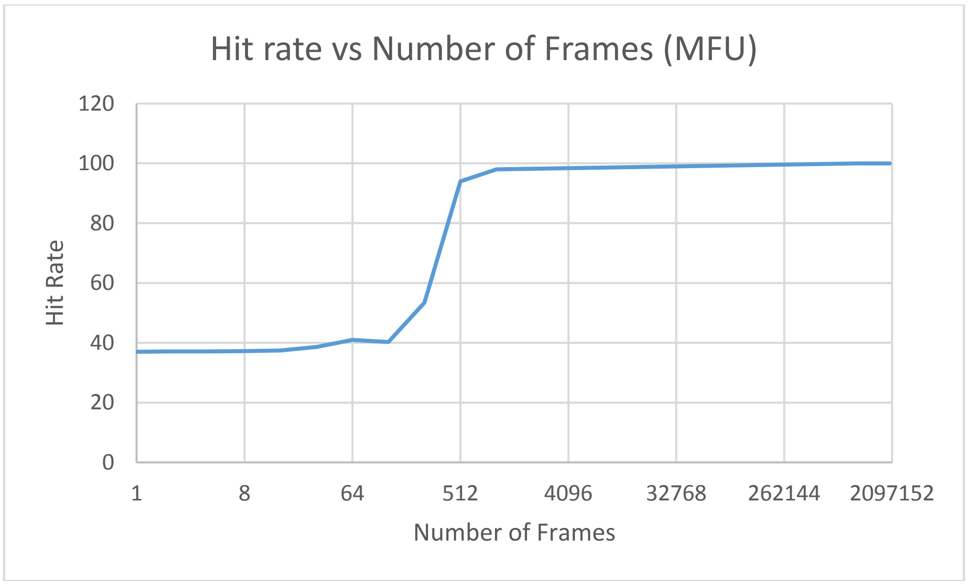
Number of frames	Hit rate
1	37.026
2	37.049
4	37.114
8	37.56
16	38.516
32	58.323
64	70.024
128	71.54
256	99.961
512	99.881
1024	99.881
1048576	99.968
2000000	99.968



LFU is not optimal very early (upto number of frames being 32). When the number of frames hits 256 the algorithm starts performing optimally. After that point the hit rate is very consistent. For this algorithm the number of frames does not matter too much since it performs relatively the same when the number of frames is smaller, larger or equal to the page size. The worst case in our trials is when the number of frames is 1.

MFU

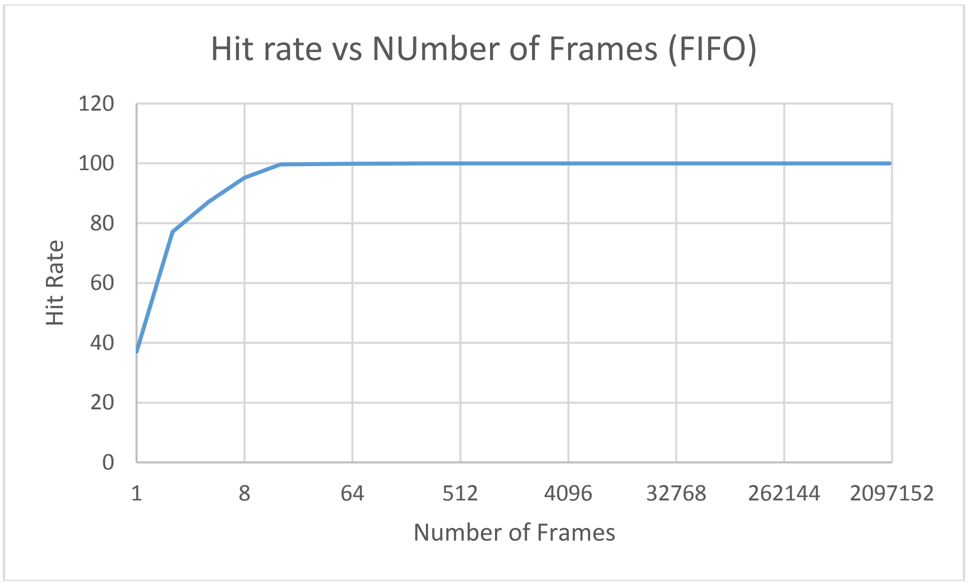
Number of frames	Hit rate
1	37.026
2	37.1
4	37.1
8	37.2
16	37.44
32	38.63
64	40.967
128	40.267
256	53.395
512	93.983
1024	97.968
1048576	99.968
2000000	99.968



MFU performs poorly very early on and starts performing optimally when the number of frames is 512. The algorithm peaks out when the number of frames is equal to the page size so increasing the number of frames past that point doesn't make vary the hitrate.

FIFO

Number of frames	Hit rate
1	37.026
2	77.116
4	87.14
8	95.217
16	99.618
32	99.75
64	99.853
128	99.911
256	99.949
512	99.968
1024	99.968
1048576	99.968
2000000	99.968



Became efficient very early on just like LRU. The hit rate is best when the number of frames is 512. After that point the hit rate peaks. Overall the algorithm is efficient when the number of frames is smaller, bigger or equal to the page size except when the number of frames is one in which case the hit rate is the same as all the other ones.

Conclusions

One interesting observation made is that no matter the algorithm when the number of frames was 1 they all had the same hit rate which makes sense for memory that can only hold one page because more frequently you encounter misses rather than hits. Each page replacement algorithm shows characteristics suited for different sizes of memory such that there is no best page replacement algorithm. Although least recently used page replacement algorithm showed exceptional performaces regardless of the size of memory whereas the random page replacement policy performed poorly no matter the size of memory.