CSCI 4707 LAB 3

ANALYSIS OF TIMES FOR LRU REPLACEMENT POLICY AND CLOCK

LRU TIME

	BUFFER TEST 1		BUFFER TEST 2	
	10k (Seconds)	**100k(seconds)**	**10k (Seconds)**	**100k(seconds)**
TRIAL 1:	0.1099	0.815853	0.442452	0.331799
TRIAL 2:	0.134116	0.61568	0.271836	0.306072
TRIAL 3:	0.58741	0.183957	0.273177	0.332499
TRIAL4:	0.119915	0.348559	0.276118	0.277082
TRIAL 5:	0.50321	0.329532	0.259342	0.458943
AVG:	**0.2909102**	**0.4587162**	**0.304585**	**0.341279**

CLOCK REPLACEMENT TIME

	BUFFER TEST 1		BUFFER TEST 2	
	10k (Seconds)	**100k(seconds)**	**10k (Seconds)**	**100k(seconds)**
TRIAL 1:	0.678118	0.522049	0.265133	0.353545
TRIAL 2:	0.295771	0.42121	0.307122	0.291356
TRIAL 3:	0.05968	0.187481	0.245917	0.348342
TRIAL4:	0.078937	0.229155	0.275348	0.243644
TRIAL 5:	0.067492	0.401482	0.324955	0.275062
AVG:	**0.2359996**	**0.3522754**	**0.283695**	**0.3023898**

According to the analysis presented we can see that the CLOCK performs faster than LRU. For the way the clock functions it replaces using two major params the usage_count(ref bit) and the ref_count(pin count) .Although it uses the ref_count to check if the pin is 0 as done in LRU, it also needs to check if the usage_count to check if its 0.usage_count acts as a clock hand and when its 0,it evicts the page alongside with when the ref_count is 0.The usage_count loops around to the beginning of memory till it gets to the end. It also scans over all the buffers in the system. It operates iteratively. For the implementation of our LRU. We utilized a queue in the form of a doubly linked list and keeping track of the previous and next pointers. If the ref_count(pin count) is 0 we add to the queue. The buffer that is stored at the head of the queue is the Least Recently Used.

A possible reason why LRU is possibly slower than clock is due to the fact that while a buffer is in the queue,this buffer can be pinned which would require the buffer to be removed and the queue would need to be re-arranged . So basically there would be a lot en-queuing and dequeuing. A buffer can get pinned and unpinned repeatedly. The continuous operation of pinning,unpinning and re queuing would become more and more expensive which would result in a slower evaluation time as seen in the analysis above. Another consideration to make is the possibility of sequential flooding according to the test data that involves range searching. A lot of misses can occur in the buffer which could also affect the time it takes LRU to effectively replace.