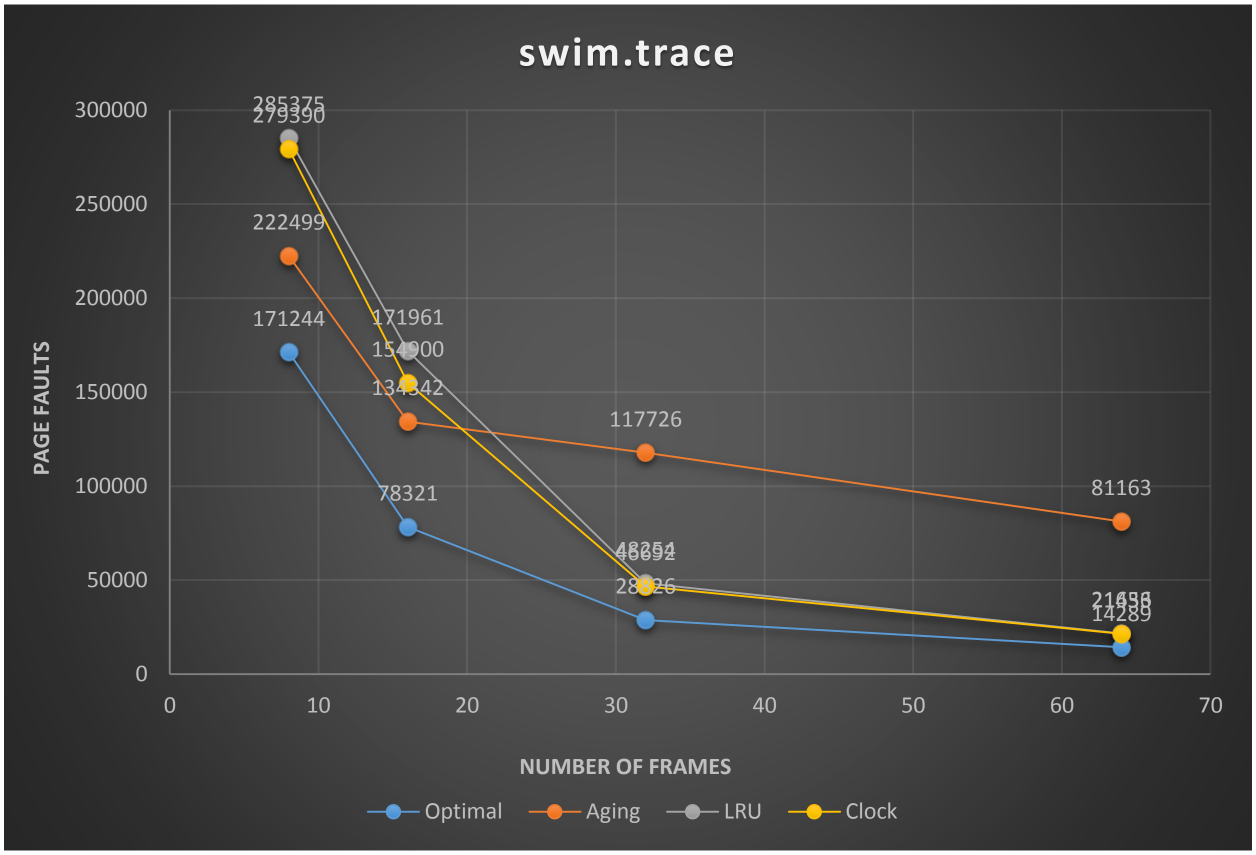
Name : You Zhou

People Soft Number : 3729901

Pitt E-mail : yoz13@pitt.edu

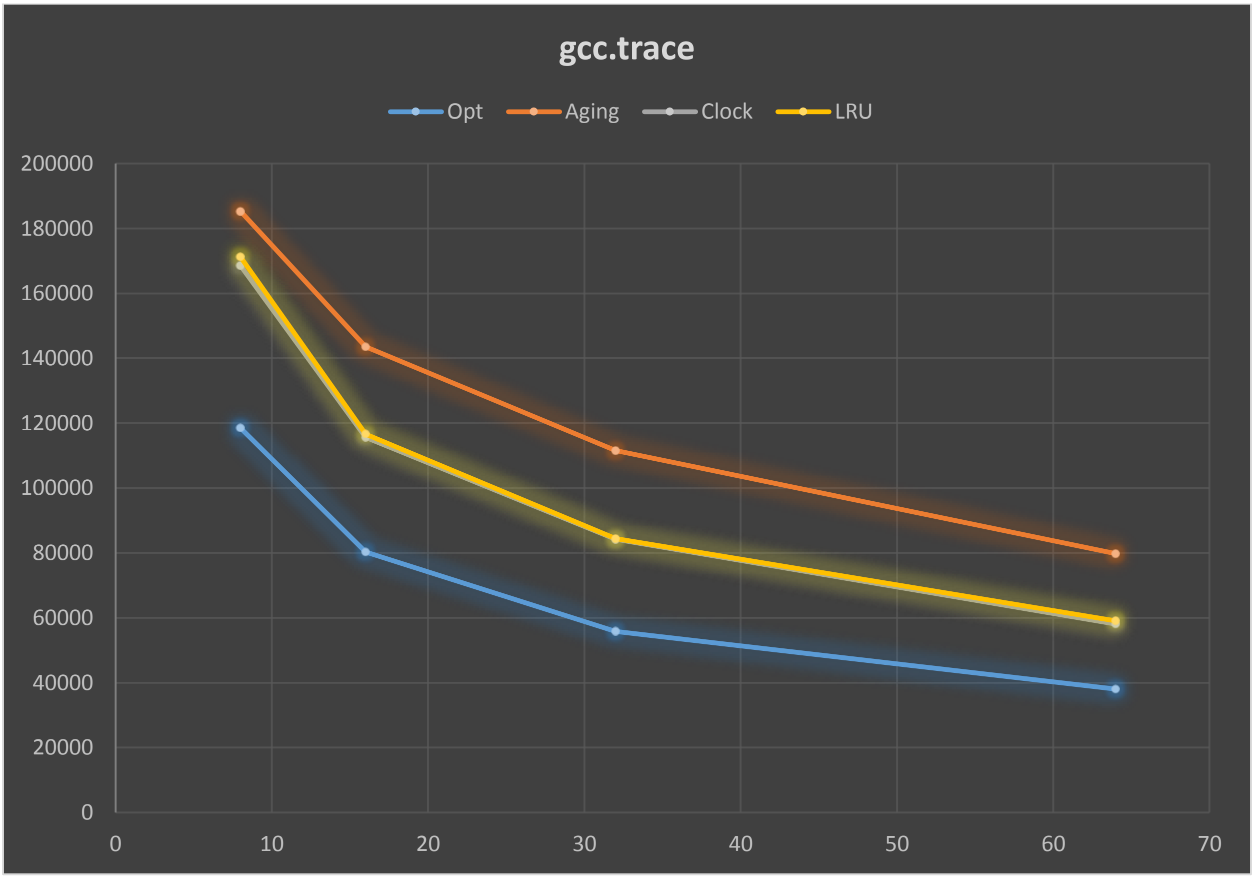
Graph – 1 : swim.trace results



Based on the graph above, none of these four algorithms suffers in **Belady's Anomaly**; in other words, they all hold the scaling property with respect to the number of frames. However, the descent rate of the aging algorithm is observably decreasing. Immediately after 16 frames, the performance began increasing in a linear manner, whereas the other three became bottlenecked only after 32 frames. Presumably, as the number of frames grows, the expressiveness of the counter degenerates. As of eight frames, 8-bit counter is expressive enough to differentiate one page from another; however, it becomes less the case when more pages are associated with, because the chance of most pages' counters concentrate within a small interval becomes likely, thereby making aging algorithm harder to approximate the more optimal page.

Starting from 32 frames, performances of LRU and Clock became indistinguishable. The underlying policy of these two are the same: evicting the least recently referenced page. They differ only on how aggressive the policy is enforced. LRU is fascinating because even if the page it puts on the top of stack is actually infrequently referenced, we do not worry weighing too much on it in a long run for if it is truly referenced infrequently, other pages will compete it down. Clock algorithm chooses to evict old unreferenced page, and, in fact, more attention is given to the oldness since very likely the case when all pages currently in memory are actively referenced, Clock algorithm degenerates to FIFO, but nearly never would LRU. The convergence of their performances suggests that oldness and unreferenceness are positively correlated.

Graph – 2 : gcc.trace results



The above discussion still holds consistently with the second graph. In this case, lines of LRU and Clock even overlapped. Also, Aging algorithm still gives out the worst performance. However, from the perspective of actual implementaion, Aging and Clock are far more superior than the other two. LRU, as mentioned earlier, extensively does stack operations, which is linear for most of time, given LRU always access arbitrary elements of the stack instead of front or back. Aging, on ther other hand, can achieve amortized theta log n in both insertion and searching with tree based data structure. Although Clock is also implemented in list based data structures like circular list, it is favored over LRU for it does not move element around – it is linear just like LRU in most operations but with a far less constant factor.

Refresh Interval for Aging

With two different trace files, its optimal settings with respect to different frames fall into intervals:

UNIT: loop iteration

8	frames →	[60 to	120)
16	frames →	[130 to	240)
32	frames →	[270 to	450)
64	frames →	[670 to	1600)

My script tested dozens of values within each interval to make a loose bound showed above. However, the general pattern is obvious to us. That is the more the frames, the somewhat quadratically longer Aging algorithm needs to wait before it can effectively tell the difference among pages.