OS - Page replacement

April 14, 2019

Group 7

Bhimavarapu Sasi Kiran - 17114021

David Gokimmung - 17114023

Gandhi Ronnie - 17114029

Suresh Babu Gangavarapu - 17114030

Goddu Vishal - 17114034

Harsha Vardhan - 17114047

P Krishna Yeswanth - 17114055

Saurabh Singh Gautam - 17114069

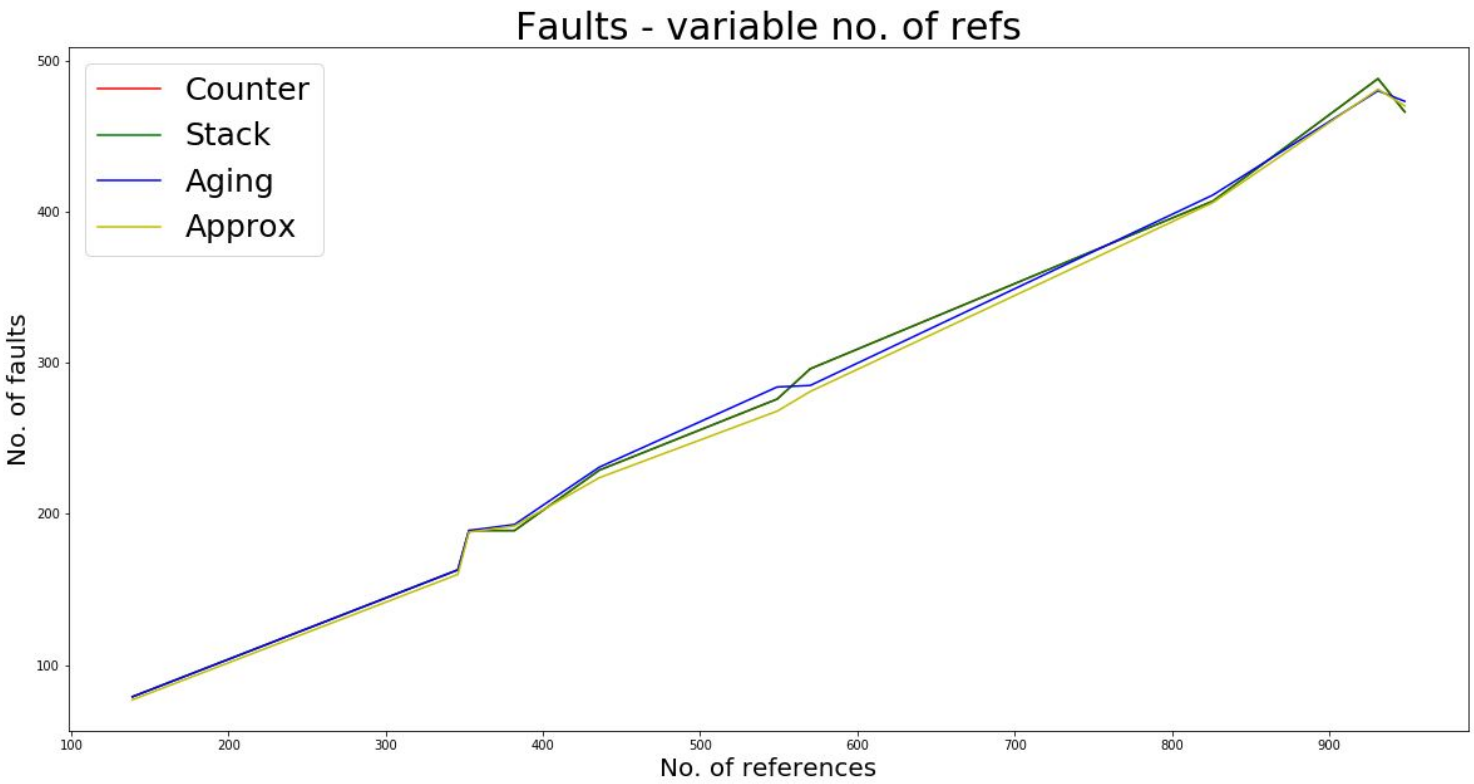
Fault graphs for various algorithms:

We have 3 variables in total, when creating our test case

1. **n_refs** - The no. of page references requested
2. **Table_size** - The size of the page table
3. **max_page** - The maximum value of the frame that can be referenced

Each of these is plotted on the x-axis and faults are plotted on the y-axis

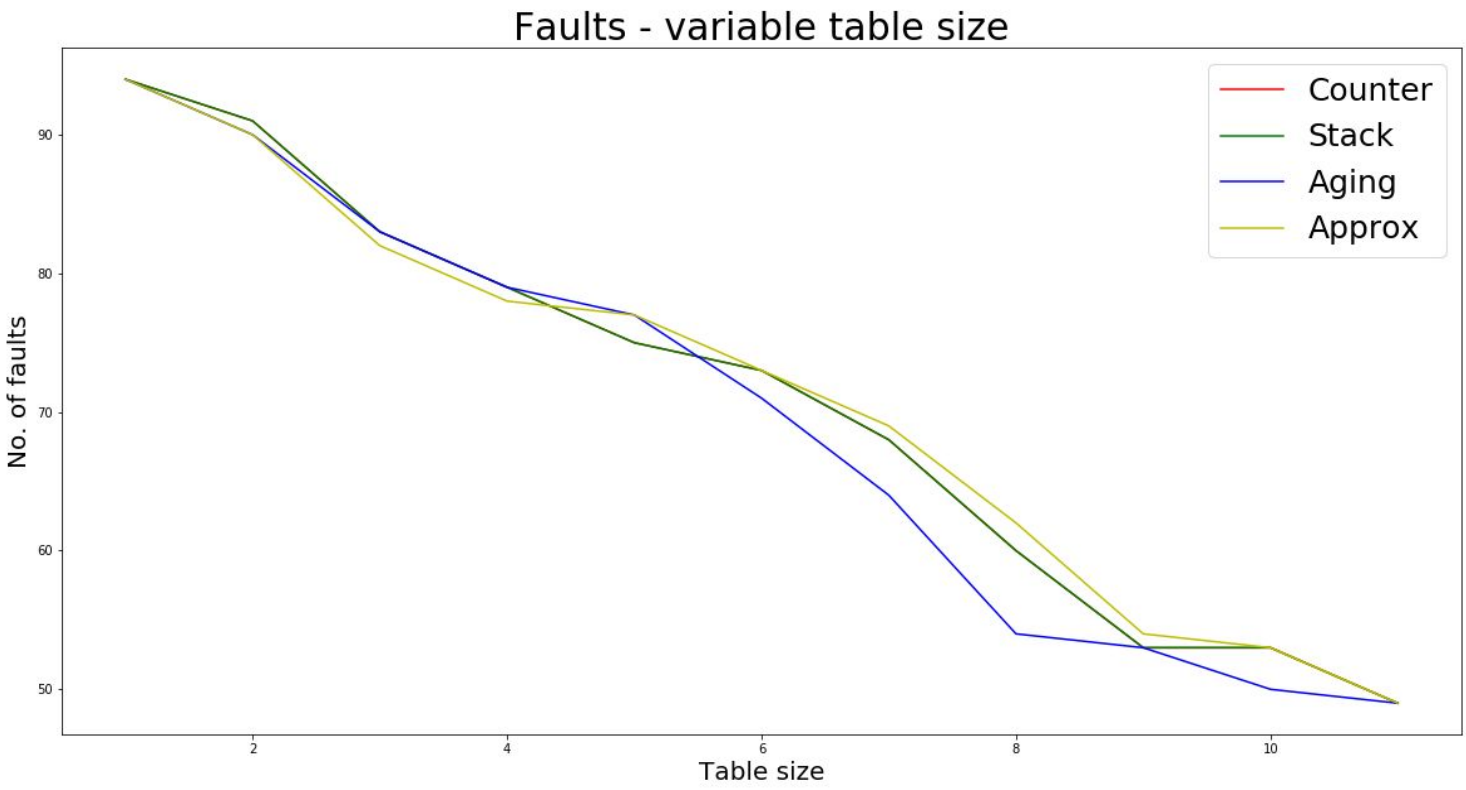
Variable no. of references



As the no. of references increases keeping other variables constant, page faults increase as we'll have to replace pages more times. Counter and Stack give the exact same results as they both replace the least recently used frame whereas, Aging and Clock algorithms approximate the least recently used frame and hence their fault count is very close to that of Counter/Stack.

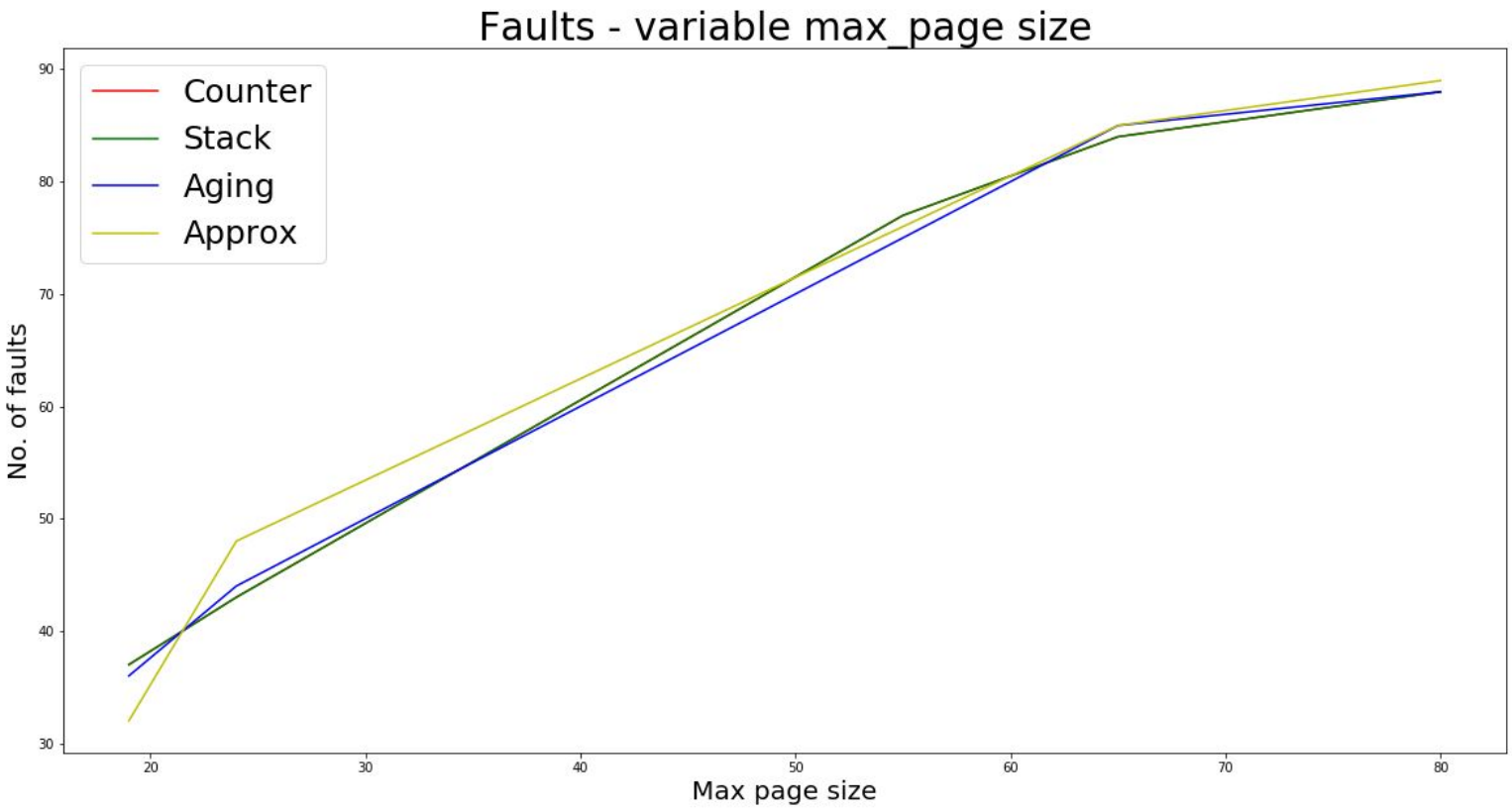
Counter cannot be seen in the graph as both Counter and Stack give the exact same results and hence Stack's graph lies exactly on top of Counter's graph and covers it.

Variable table size



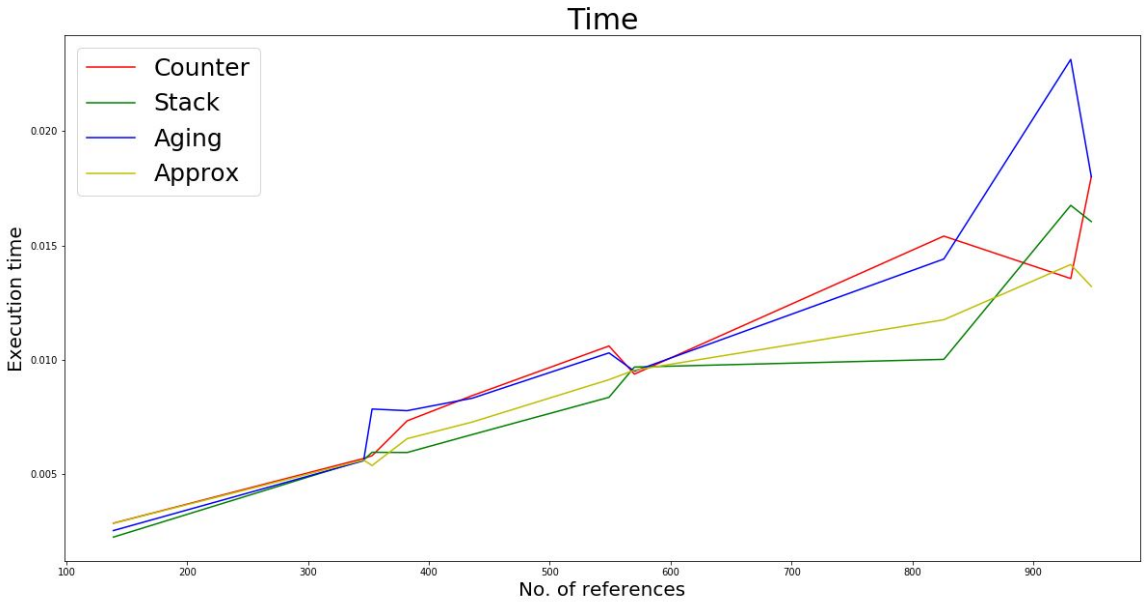
With the increasing table size, we will have more slots to put our frames in and hence resulting in a decrease in the fault count. Counter and Stack will have the same graph in this case as well as they replace the exactly LRU frame and aging and clock method do a good job at approximating, again.

Variable max_page

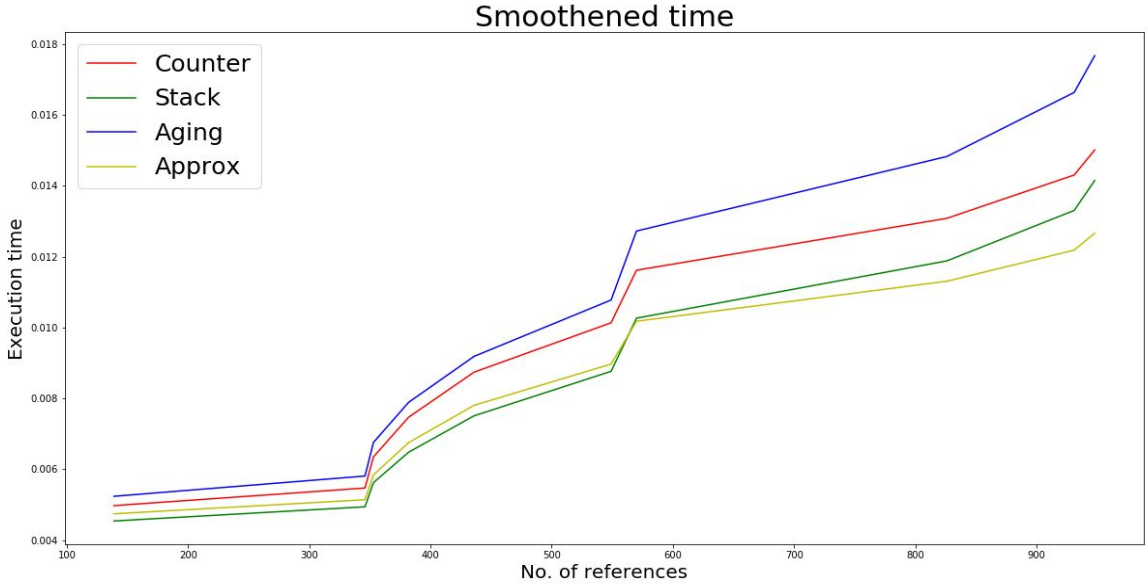


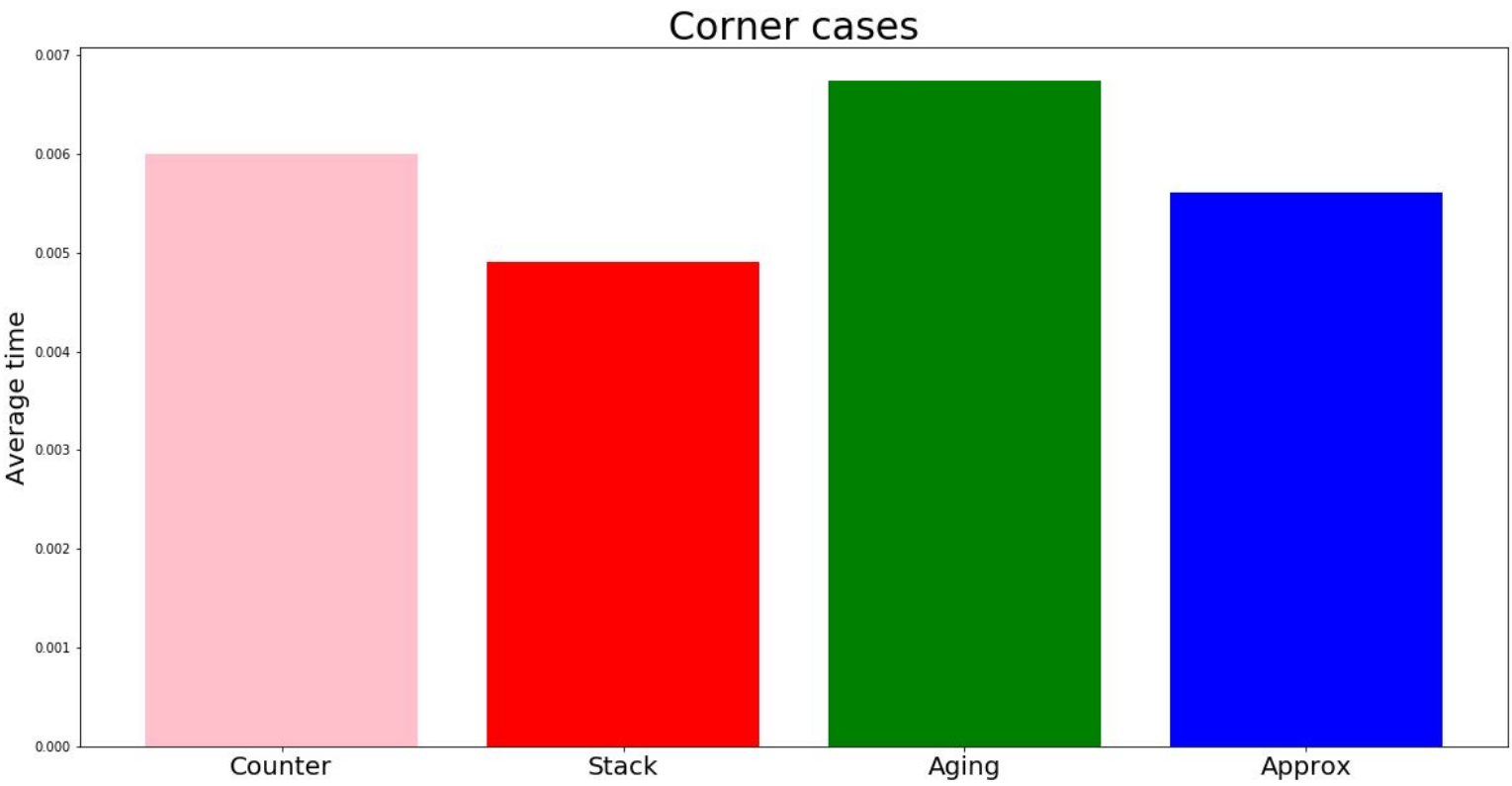
As max_page increases, there exists more variety in the referenced frames and hence we'll have to replace other frames from the page table more often. Therefore, no. of faults decrease with increasing max_page. Again, Aging and Clock algorithms do a good job of approximating Counter and Stack algorithms.

Execution time



Although we can get an idea about the relative running times of each algorithm from this graph, it is hard to infer data from it. Therefore, we ran a Gaussian filter over our input data to smoothen the graph. As expected, Aging and Counter methods take the most time.





For corner cases, we tested a few inputs that produce page faults for every frame referenced and averaged the results into a bar plot. Aging takes the most time as in every step of our process the iterator will have to traverse the whole table and to find which element is the oldest and also right shift every counter in the page table every step, whereas in counter we will only have to increment the specific element. Approx will take lesser time as all data elements are given a single boolean bit. Stack will take the least time as the complexity is constant and you have to only pop the first element when a page fault occurs.

Complexity Analysis:

Stack:
The best case and the worst case will have fixed time complexity as we only have to pop the first element from the stack. We know exactly what to do when a page fault occurs, without having to iterate through the stack.

Counter:
The best case will be O(1) when our LRU is in the first spot of the table and worst case will be O(table_size) when it is in the end of the page table, as we have to search for the page with the smallest counter throughout the table. (Assuming that table is implemented as a normal array)

Aging:
The best case will be O(1) when our oldest page is in the first spot of the table and worst case will be O(table_size) when it is in the end of the page table, as we have to search for the oldest page throughout the table. (Assuming that table is implemented as a normal array)

Approx:
The best case will be O(1) when the page with 0 bit is in the same spot as the traversing pointer of the table and worst case will be O(table_size) when it is exactly above our traversing pointer.

Comments:

Aging and clock method gave very similar page faults thus fitting the definition of approximation. Aging is limited by the no. of bits we use to keep track of age. We used ‘int’ data type to store our bits, hence our algorithm was only able to keep track of the last 32 iterations. As seen in graph, counter and stack always have the same number of faults as both of them follow the LRU algorithm whereas in terms of speed, stack outperforms all other methods as in the stack method only the first page needs to be popped in the case of a page fault and the worst time would be that of aging as not only do we have to right shift every counter in the page table, but when a page fault occurs, we have to search the entire table for the most aged counter.