Assignment 5 : Virtual Memory management Simulator

Design Document

15CS10051
15CS10019

Functions in the program -

The program consists of functions 1
for loading a page to the memory(add_page()) ,
finding and removing the victim page from memory (get_victim()),
resetting the ref bit to zero for all pages and moving corresponding pages from one class to other when using NRU (reset_ref()) ,
reading input and simulating the MMU (simulator()),
generating statistics of the MMU performance (print_usage()) and
for printing the page table for the process (pagetable_print()).

The usage of program is as follows

```
g++ -o simulate Ass5_16.cpp -std=c++11
./simulate < test.txt
```

Ass5_16.cpp is the source file and test.txt contains the instructions.

The test data can be generated as follows

```
g++ -o data test.cpp -std=c++11
./data <pages> <no_of_instr> <working_set_size>
<probability(%)from_working> > output.txt
```

The data is generated and stored in the output.txt
Or we can just use the make file

```
make all
```

Structure -

FIFO and Second Chance FIFO -

Both of these algorithms are implemented using a single queue data structure .

When add_page() is called that page is added to queue and for the get_victim() the page at the top of the queue is popped and returned.
For second chance an extra step of checking the reference bit is present.

Random -
In these a vector of page numbers is maintained and whenever get_victim is called a random page from these vector is returned.

LRU -
For these a separate doubly linked list data structure is created from the struct nodes and a corresponding hashmap is maintained for referencing these nodes directly in the list.
Whenever add_page() is called the corresponding node is added to the top list.
whenever usedit() is called the corresponding node of the page is removed from the middle of the list and added to the top.
Whenever get_victim() is called the page corresponding to the bottom node of the list is returned and it is removed from the list

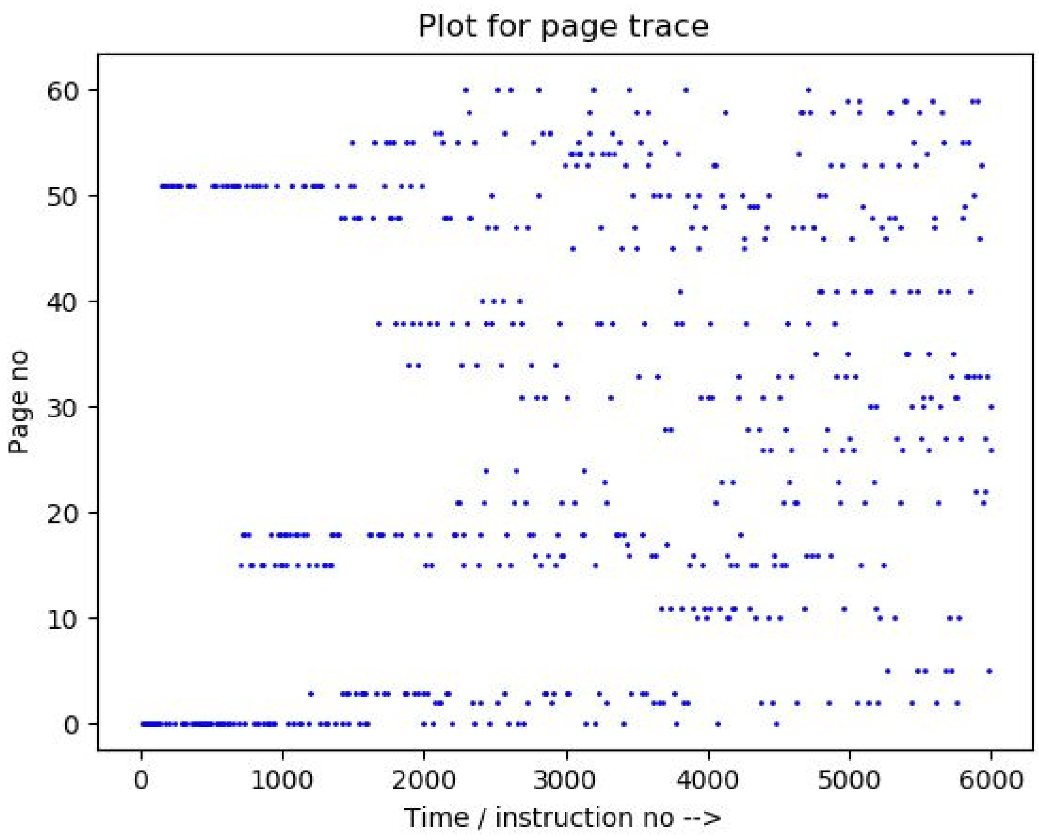
NRU -
For these four vectors are maintained for each class in NRU and these are updated when get_victim(),add_page() or moveto() is called based on the NRU page replacement algorithm.

Experimental Results -

The graph corresponding to the every 10 th page trace consisting of 6000 instructions , with a 99% probability of selecting for the working set and a working set size of 100 .

The trace.txt file contains page trace generated by the data.cpp program with following parameters .
no of instructions = 1000000
working set size is = 1000
probability that next page is from working set is = 99%

```
./data 64 1000000 1000 99 > trace.txt
```


Plot for page trace
Page no
Time / instruction no -->
0
10
20
30
40
50
60
0
1000
2000
3000
4000
5000
6000