Santa Clara University

Department of Computer Engineering

Advanced Operating Systems (COEN 383)

Project-4 (6 pts)

Group 3
Sruthi Thiyagarajan
Mitali Sahoo
Bharat Chadalawada

Swapping and Paging

We built a simulation written in C++ programming language that experiment with multiple processes running concurrently, each process start at page-0 then every100 msec it references a random page from its own address space taking into consideration the locality of reference algorithm as described in the Homework assignment.

Locality of reference, after referencing a page i, there is a 70% probability that the next reference will be to page i, i-1, or i+1. i wraps around from 10 to 0. In other words, there is a 70% probability that for a given i, Δi will be -1, 0, or +1. Otherwise,|Δi| > 1.

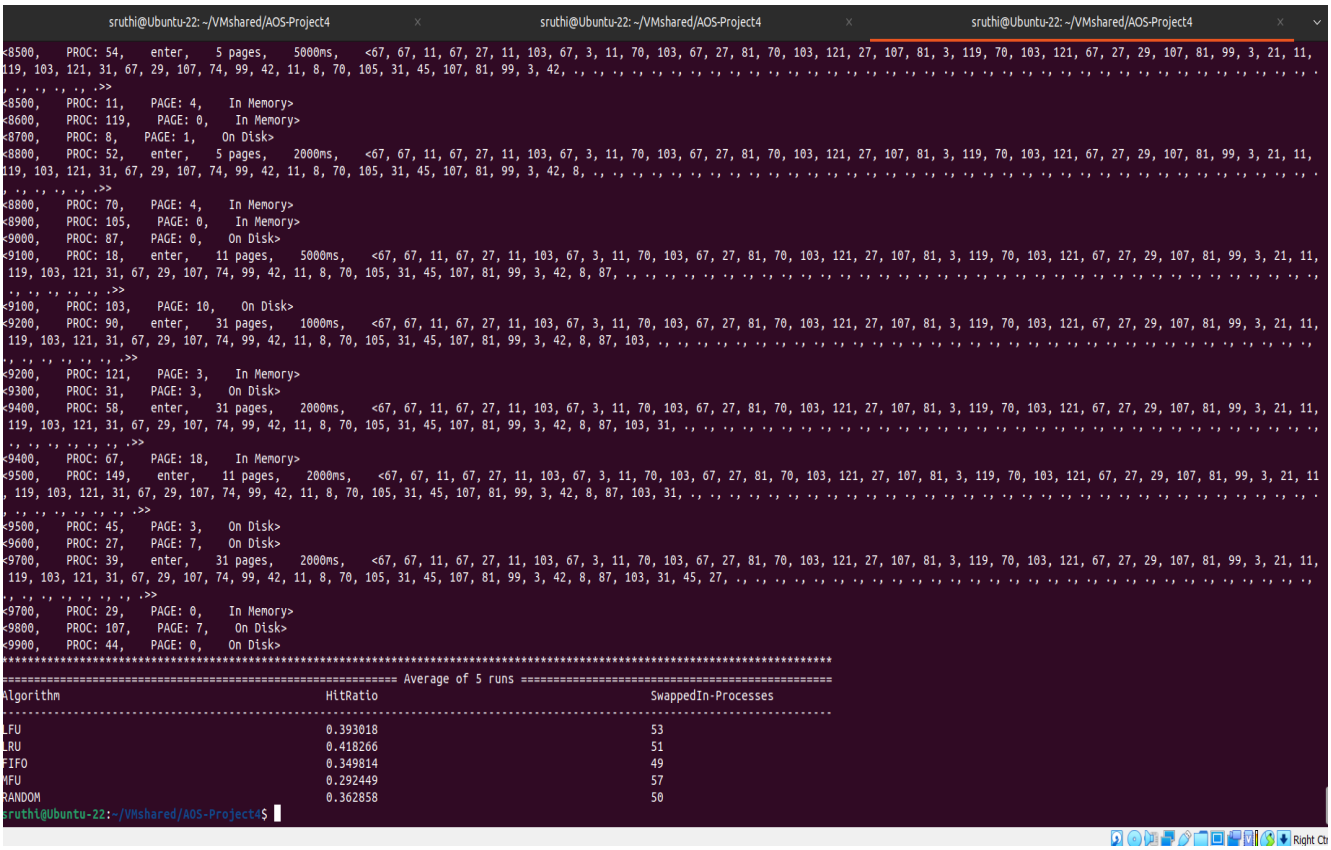
Workload Generation

We generated 150 jobs <process Name, Process size in pages, Arrival time, Service Duration>. We Sorted the random jobs generation based on arrival time and structured them as vectors. Processes have randomly and evenly distributed sizes of 5, 11, 17, and 31 MB. Processes have randomly and evenly distributed service durations of 1, 2, 3, 4, or 5 seconds.

Simulator:

i. Generated the workload and represent it as sorted queue based on arrival time
ii. Created and initialize the free page list, initially with 100 pages, each is 1 MB.
iii. Picked up one job at a time from the Job queue and if there are 4 free pages in the free page list then start running that process, otherwise wait till one of the existing processes complete.
iv. Generated the appropriate record whenever starting or completing a job <time stamp, process name, Enter/exit, Size in Pages, Service Duration, Memory-map>.
v. Once a job started execution, it generates a memory reference every 100 msec to a random page from its own virtual address space; need to generate an appropriate record <time-stamp in seconds, process Name, page-referenced, Page-in-memory, which process/page number will be evicted if needed>.
vi. If memory is all used and process reference a page that is not currently in memory then we need to apply the chosen “page replacement Algorithm” to select a victim page to evict so you can bring to memory the needed page.
vii. We ran the simulator 5 times, each is 1 minute, and each time using different replacement algorithm (algorithms FIFO, LRU, LFU (Least Frequently Used), MFU, and random pick).
viii. We continued running until the 1 minute expires, collect and save the requested statistics and exits.
ix. We also ran simulator 5 times, each to complete the one minutes, and compute the hit/miss ratio of pages referenced by the running jobs for each run. Then get average of 5 runs.
x. We Ran the simulator for 100 page references, and for each reference, print the <time-stamp in seconds, process Name, page-referenced, if-Pagein-memory, which process/page number will be evicted if needed>.
xi. For each replacement algorithm, print the average number of processes (over the 5 runs) that were successfully swapped-in (Process execution started).

The screenshots of the output for our Project have been pasted below:



The best performance in terms of Hit Ratio - we got is Least Recently Used Algorithm.