**CSCI 5103 Operating System Project 3**

**Group Member**

Yutong Zhao 5593405

Qi Le 5674954

● In your own words, briefly explain the purpose of the experiments and the experimental setup. Be sure to clearly state on which machine(s) you ran the experiments, and what your command line arguments were. So that we can reproduce your work in case of any confusion.

The purpose of the experiments is to compare difference page replacement algorithms under different benchmark programs and varying number of frames with fixed 100 pages when the number of pages in larger than the number of frames. The page replacement algorithms include Rand, Fifo and Custom. The benchmark programs include Scan, Focus and Sort. We test all combinations of different page replacement algorithms and benchmark programs under 1, 10, 30, 50, 80, 99 and 100 frames with fixed 100 pages

We run the code in the Linux System. The command line would be the same as the instruction in the hw3.pdf: $ ./virtmem [npages] [nframes] [rand|fifo|custom] [scan|sort|focus]

● Describe the custom page replacement algorithm that you have implemented. Make sure to give enough details that someone else could reimplement your algorithm without your code.

We named the custom page replacement algorithm Dynamic Adjustable Frame Allocation (DAFA).

We initialized the number of evictions of each frame with 0 (*DAFA\_array*). Then, DAFA evicts the frame that has been least evicted when the eviction is needed. After the eviction, the corresponding number of evictions of the evicted frame would increase by 1. We also construct a temp array (*new\_DAFA\_array*) to store the number of evictions of each frame during each period. At the next period, the *DAFA\_array* would be updated by the *new\_DAFA\_array*. The period is determined by the base number and the ratio of number of pages and the number of frames. We set the base number to 100. When the ratio is high, the collisions would happen a lot. So the period would be short to let the program adjust the frame eviction. When the ratio is low, the collision would happen less. So we can extend the period and decrease the update frequency.

● Measure and draw graphs of the number of page faults, disk reads, and disk writes for each program and each page replacement algorithm using 100 pages and varying numbers of frames between 1 and 100. Spend some time to polish your graphs such that they are nicely laid out, correctly labelled, and easy to read.

**SCAN**

*Figure 1. Page Faults Number vs Number of Frames using different page replacement algorithms and scan benchmark program*

*Figure 2. Disk Reads Number Number vs Number of Frames using different page replacement algorithms and scan benchmark program*

*Figure 3. Disk Writes Number vs Number of Frames using different page replacement algorithms and scan benchmark program*

**FOCUS**

*Figure 4. Page Faults Number vs Number of Frames using different page replacement algorithms and focus benchmark program*

*Figure 5. Disk Reads Number vs Number of Frames using different page replacement algorithms and focus benchmark program*

*Figure 6. Disk Writes Number vs Number of Frames using different page replacement algorithms and focus benchmark program*

**SORT**

*Figure 7. Page Faults Number vs Number of Frames using different page replacement algorithms and sort benchmark program*

*Figure 8. Disk Reads Number vs Number of Frames using different page replacement algorithms and sort benchmark program*

*Figure 9. Disk Writes Number vs Number of Frames using different page replacement algorithms and sort benchmark program*

● Analyze your results and describe when one algorithm outperforms the others, and why

**SCAN:**

First, when the number of pages equals to the number of frames, all the page replacement algorithms have the same results of the page fault number, the disk reads number and the disk write number.

Second, FIFO is the worst page replacement algorithm in scan benchmark program. The reason would be that we manipulated the data sequentially and the page fault would happen every time when the scan benchmark program switches page.

Rand has least written faults in the whole writing test and the least read faults and the page faults in the first half period of testing. The frames have the same probability to be evicted in rand, so when the number of frames increases, the number of faults decreases.

DAFA has the least read faults and the page faults in the second half period of testing. Our dynamic allocation works well when the updating period is not too short, and it performs pretty good in scan. The reason would be that the updating *new\_DAFA\_array* let the program choose the frame thast has been evicted least in the previous period to evict. This is kind of LRU. But for written faults, since the number of write operations is small, the DAFA would act as the FIFO.

**FOCUS:**

For focus benchmark program, we found that rand is the best page replacement algorithm and the fifo is the worst page replacement algorithm.

Focus benchmark algorithm replaces the number in data randomly. We can see that all the numbers of faults decrease as the numbers of frames increase because the probability of replacing the data in the page that has been evicted decreases and the page faults decrease.

Moreover, the seed remains the same for the whole focus program, this may lead to the rand is the best page replacement algorithm. If we set the seed to be randomly chosen, the result might change.

**SORT:**

For sort benchmark program, all page replacement algorithms have the same performances. They all have a huge decrease when the numbers of frames are 10. There are two reasons. First, the seed is small, and it keeps the same, so the data would be the same. Second, the qsort() function uses the quicksort as its algorithm. Quicksort is a divide and conquer algorithm. When the number of frames is small, we would trigger the page fault many times. Once the number of frames hits 10, it would be enough for the quicksort.

**Summary:**

In the three tests, rand would be the best page replacement algorithm. But the randomness in this experiment is huge and we do not proceed the multiple times of the experiments. The random seed keeps the same in all the experiments might affect the results. Our custom algorithm performs better than or close to the fifo in all the situations, which is unexpected and gratified.