# CMPS 111 — Spring 2017 – ASGN3

Alan Duncan, Aryan Samuel, Khoa Hoang

## Description

The goal of this project is to experiment with the FreeBSD pageout daemon. The purpose of these modification is to analyze the difference of our modifications to the paging algorithm to our modified Slim Chance Algorithm.

The Slim Chance algorithm puts the inactive and invalid pages on the rear of the free list and instead of subtracting from the activity count we will divide it by two and move it to the front of the active list. Then finally when we move a page to the inactive list we will move it to the front.

Files Modified in Assignment:

- vm_pageout.c
- vm_phys.c
- vm_page.c

## Statistics Generated:

All of our statistical data was gathered inside the vm_pageout_scan() function inside the vm_pageout.c file. The counters we used to keep track of our data was:

- The total number of pages in each queue
- The number of pages moved from Active to Inactive
- The number of pages moved from Inactive to Cache
- the number of pages moved from Inactive to Free
- The number of pages queued for Flush

Each individual counter keeping tracking of the number of pages gets incremented in their proper position which is detailed within our file modification section.

## vm_pageout.c

For this c file, we are adding several variables to keep track of the number of pages being moved around, which is more detailed above. Most if not all of our code to log the number of pages is located inside the $vm\_pageout\_scan()$ function.

We first begin by including the proper syslog.h library to print the statistical data to the log file in var/log. We then declared our variables inside $vm\_pageout\_scan()$ as this is where each data point is handled in a way we felt is appropriate for us to gather properly for logging and statistical analysis.

The total number of pages in each queue is incremented for each iteration of the inactive queue. This only accounts for half the total number and the variable will again be iterated through each iteration of the active queue as well.

The assignment also specifies that we must modify how we handle the activity count: divide by 2 instead of subtracting. This is handled inside the loop for the active queue.

The number of pages moved from inactive to free and inactive to cache is incremented in this same loop as well since its the inactive queue loop that scans and checks if pages can be moved to either cache or free, as well we check in here for the number of pages queued for flush

**vm_pageout.c**

**Algorithm 1** vm_pageout_scan()

```
include sys/syslog.h
...
for Pages in Inactive Queue do
    ...
    Total number of pages in each queue+1
    ...
    if Page is valid then
        ...
        Number of pages moved from Inactive to Free+1
        ...
    end if
    if Page is Dirty then
        ...
        Number of pages moved from inactive to cache+1
        ...
    end if
    if vm_pageout_clean(m) ! = 0 then
        ...
        Number of pages queued for flush+1
    end if
end for
...
for Pages in Active queue do
    ...
    Total number of pages in each queue+1
    ...
    if not act delta then
        Divide activity count by two
    end if
end for
```

**vm_pageout_init()**

Here we changed the time on which pages are scanned. So, the default value was originally set to 600 seconds, we changed this to 10 seconds as specified in the assignment. This change will cause the number of pages being scanned to increase during each pageout.

**Algorithm 2** vm_pageout_init()

1: **if** Page update period is 0 **then**
2: Time on which pages are scanned to 10 seconds
3: **end if**

## vm_phys.c

Here we are modifying one function named $vm\_phys\_free\_pages()$ to place pages on the front of the free list queue rather than the back. We are simply changing the default value this function takes when invoked from 1 to 0, as 1 signals adding it to the front and 0 to the back.

**Algorithm 3** vm_phys_free_pages()

1: vm_freelist_add(fl, m, order, 1) - change 1 to 0

## vm_page.c

For this file, we are modifying some functions to place inactive pages at the front of a list instead of the rear. We are simply modifying $\_vm\_page\_deactivate()$ and $vm\_page\_deactivate()$ so that it can simply add to the rear of the list. To be more specific, we are changing the default values that the function takes in from 0 to 1. The reasoning for this because FreeBSD has built-in mechanisms that reads an int value for pages and then that tells whether pages should be inserted at the head or tail. Not only that, we are also removing the option of adding a page to the head of a free list by modifying how the core code works, the pseudo-code goes into more detail about this.

More specifically, we change TAILQ_INSERT_HEAD to TAILQ_INSERT_TAIL so that in case if the default value somehow still does not insert to the tail, it will force the page to insert at the tail regardless. We do this for several other functions described in the pseudo-code below.

**Algorithm 4** vm_page_deactivate()

1: _vm_page_deactivate(m, 0) - change 0 to 1

**Algorithm 5** _vm_page_deactivate()

1: ...
2: TAILQ_INSERT_HEAD to TAILQ_INSERT_TAIL

**Algorithm 6** _vm_page_requeue_locked()

1: ...
2: TAILQ_INSERT_HEAD to TAILQ_INSERT_TAIL

**Algorithm 7** _vm_page_requeue()

1: ...
2: TAILQ_INSERT_HEAD to TAILQ_INSERT_TAIL