Members: Mackenzie Huynh (Captain)
Anthony Lin,
Khachatur Mirijanyan,
Priya Calaimany
Team: Evanescence
Date: March 19, 2019

DESIGN DOCUMENT

Task:

The purpose of this project is to alter the page placement algorithm and the activity rate of the pageout daemon in order to compare the page fault rates of the BON CHANCE algorithm and the CLOCK algorithm.

Basic Terms and Definitions:

Types of Scheduling Algorithms:

Q: What is the second chance algorithm?
A: The second chance algorithm is a page replacement algorithm in which the reference bit of the oldest page in memory is inspected. If it is 0, the page is replaced immediately. If it is 1, the bit is cleared and the page is put onto the end of a list of pages.

Q: What is the clock algorithm?
A: The clock algorithm is a page replacement algorithm in which the reference bit of the oldest page in memory is inspected. If a page is found to be referenced in both this scan and the previous scan, it is moved to the lower queue (active queue). If a page is found to be referenced in neither this pass nor the previous pass, the page is moved to the higher queue (inactive queue).

Q: What is the bon chance algorithm?
A: The bon chance algorithm is a page replacement algorithm in which inactive and invalid pages are placed in the free list based on their page numbers, pages placed in the active list are assigned random activity counts between 1 and 32, and pages placed in the inactive list are assigned random activity counts between 1 and 10. Pages are placed in the active and inactive lists based on their activity counts.

Types of Data Structures:

Q: What is a *vm_page* (see vm_page.h) data structure?

Members: Mackenzie Huynh (Captain)
Anthony Lin,
Khachatur Mirijanyan,
Priya Calaimany
Team: Evanescence
Date: March 19, 2019

DESIGN DOCUMENT

A: This data structure represents a page in the FreeBSD kernel. It is usually referenced as a vm_page_t = *vm_page. It contains all the page information, including activity count and physical address. Activity count is important, as this value is used to decide whether to move a page from one queue to another.

Q: What is a *scan_state* (see vm_pageout.c) data structure?
A: This data structure records data during a run of the pageout daemon. It records how many pages have been scanned, etc.

Requirements:

1. Random Activity Count Assignment:
    a. Initially assign each page a random activity count between 1-32 when it is placed in the ACTIVE queue.
2. Page Number Based Queue Position Assignment
    a. Insert even-numbered pages at the front of the free list and odd-numbered pages at the rear of the free list.
3. Increased Scan Rate
    a. Change the scan rate of the pageout daemon so that it runs faster than every 10 minutes (its current rate).

Programming Decisions:

Observations/Study of Original Kernel Code:

vm_pageout.c:

- *struct scan_state* contains a mechanism for recording data while scanning a page queue.
- The functions *vm_pageout_init_scan* and *vm_pageout_end_scan* initialize the page scan count to 0 and record the final page scan count, respectively. *vm_pageout_collect_batch* updates the page scan count.

Members: Mackenzie Huynh (Captain)
Anthony Lin,
Khachatur Mirijanyan,
Priya Calaimany
Team: Evanescence
Date: March 19, 2019

DESIGN DOCUMENT

- *vm_pageout_update_period* is set to check pages at a rate of 600 seconds. 600 seconds / 60 seconds = 10 minutes. This is what the assignment means when it says the current pageout daemon algorithm runs every 10 seconds.

vm_page.h:

- Here is where ACT_INIT is defined to be 5.

vm_page.c:

- Each page has an activity count. In vm_page_activate, the activity count variable in the page structure is set to ACT_INIT, thus the activity count of each page starts at 5.
    - In the original algorithm, the activity counts can increase until they reach a maximum value of 64.
    - In this assignment, initially assign each page a random activity count between 1-32 when it is placed in the ACTIVE queue.

Kernel Changes:

This table describes the changes that need to be made to specific, existing files/functions in the FreeBSD kernel and the reasonings behind those modifications.

File	Changes	Reasoning
vm_page.c	vm_page_activate(): a. Generate a random number between 1 and 32. Do this using the formula random = (arc4random()%(end-front))+front, where front = 0 and end = 33. Set *act_count* field of *vm_page_t* to the random number.	a. Changes the original activity count to a random number if entering the active queue. b. The flag is set to decide whether the page is moved to the front or back of the inactive queue. If (r > 7) the flag is set, then it gets put at the front.

Members: Mackenzie Huynh (Captain)
Anthony Lin,
Khachatur Mirijanyan,
Priya Calaimany
Team: Evanescence
Date: March 19, 2019

DESIGN DOCUMENT

	vm_page_deactivate(): a. Generate a random number between 1 and 10 using the formula random = (arc4random()%(end-front))+front, where front = 0 and end = 11. b. Set PGA_REQUEUE_HEAD flag if random number > 7.	
vm_pageout.c	vm_pageout_scan_active(): a. Divide *act_count* by 2 before decrement. vm_pageout_init(): b. Change *vm_pageout_update_period* from 600 to 10. vm_pageout_worker(): c. It logs information before and after the active_scan. Afterwards, calls dumpStats() to log statistics. dumpStats(): d. Increment counters to keep track of the required statistics which will also be logged into debug.log	a. *act_count* was already being decremented. We first divide the value by half before the decrement and before the page is moved to the front of the active queue. b. By changing the initial update period, we can get the pageout daemon to run more often. This purpose of this is to observe and analyze the effects of the BON CHANCE algorithm. c. Logs the page count in the Active, Inactive, Laundry, and Free list before active_scan. Then, calls dumpStats() afterwards to log required statistics. d. Log statistics, then resets all counters and variables. *Logging is done when benchmark is running.

Members: Mackenzie Huynh (Captain)
Anthony Lin,
Khachatur Mirijanyan,
Priya Calaimany
Team: Evanescence
Date: March 19, 2019

DESIGN DOCUMENT

vm_phys.c	vm_free_list_add(): a. Right shift *phys_addr* field of *vm_page_t* by 12. b. Mod the shifted value by 2. c. If value is 0, put the page at the head of the free list. Otherwise, it goes to the tail.	a. The page size is 4K. Thus if we right shift by 12, we get the page number (page number = physical address / page size). b. Decide if the page number is even or odd. c. Place the page in the appropriate place on the free queue based on the BON CHANCE algorithm.
Makefile		a. This file copies the modified files into /sys/vm, rebuilds and installs the kernel, then reboots the system.

Here is a more explicit summary of the functions it is necessary to modify to implement the BON CHANCE algorithm. We created and used a system variable called **use_bon_chance** inside the kernel, which we can access by calling systl and storing the value into a variable called ***isBonChance***. The variable is used to switch between the paging daemon algorithm done by FreeBSD and our implementation, Bon Chance.

Logging: (*vm_pageout.c*)

In order to record our results, we created 5 original variables:

Original Variable	Purpose
scanned	number of pages scanned in that pageout
active_to_inactive_counter	number of pages from active queue to inactive queue

Members: Mackenzie Huynh (Captain)
Anthony Lin,
Khachatur Mirijanyan,
Priya Calaimany
Team: Evanescence
Date: March 19, 2019

DESIGN DOCUMENT

inactive_to_active_counter	number of pages from inactive queue to active queue
inactive_to_free_counter	number of pages from inactive queue to free list
active_scan_occured / inactive_scan_occured	is set whenever a scan actually occurs in the queues respectively.

Function Modifications:

vm_page.c (*modified file*)

Prototype: void vm_page_activate (vm_page_t m)
Parameters: vm_page_t m (current page)
Output: nothing (void)
Return values: nothing
Description: Modify this function so that the activity count of a page is set to a random number between 1 and 32, instead of the ACT_INIT default value (5), before it is inserted into the active list.

Prototype: void vm_page_deactivate (vm_page_t m)
Parameters: vm_page_t m (current page)
Output: nothing (void)
Return values: nothing
Description: Modify this function so that the activity count of a page is set to a random number between 1 and 10 before it is inserted into the inactive list. If the random number is greater than 7, then set a flag so that the page is inserted at the front of the inactive list.

vm_pageout.c (*modified file)*

Prototype: static void vm_pageout_scan_active (struct vm_domain *vmd, int page_shortage)

Members: Mackenzie Huynh (Captain)
Anthony Lin,
Khachatur Mirijanyan,
Priya Calaimany
Team: Evanescence
Date: March 19, 2019

DESIGN DOCUMENT

Parameters: struct vm_domain *vmd (current domain), int page_shortage (number of pages to move from active queue to inactive/laundry queue)
Output: nothing (void)
Return values: nothing
Description: Modify this function so that the activity count of a page is divided by two before it is decremented and before it is placed at the front of the active list.

Prototype: static void vm_pageout_init(void)
Parameters: nothing (void)
Output: nothing (void)
Return values: nothing
Description: Modify the variable *vm_pageout_update_period* in this function so that the pageout daemon checks for page faults at a faster rate than every 10 minutes.

vm_phys.c (*modified file)*

Prototype: static void vm_freelist_add (struct vm_freelist *fl, vm_page_t m, int order, int tail)
Parameters: struct vm_freelist *fl (pointer to free list), vm_page_t m (page being inserted into freelist), int order (index of that page's buddy queue), int tail (value that determines whether page is inserted at head or tail)
Output: nothing (void)
Return values: nothing
Description: This function inserts a page at the head or tail of the free list based on page number. Modify this function so that even-numbered pages are inserted at the front of the free list and odd-numbered pages are inserted at the rear of the list.

benchmark.sh *(created file)*

Description: Runs the stress tester and outputs relevant vmstat statistics.

switch.sh *(created file)*

Description: It will switch between the Paging algorithms ./switch 0 for the current Paging done by FreeBSD or ./switch 1 to use Bon Chance, our implementation. Initially, the algorithm is doing the paging done by FreeBSD.

Members: Mackenzie Huynh (Captain)
Anthony Lin,
Khachatur Mirijanyan,
Priya Calaimany
Team: Evanescence
Date: March 19, 2019

DESIGN DOCUMENT

Makefile *(created file)*

Description: Copies relevant files into the kernel that were changed. Then, it builds and installs kernel. After, it will reboot the system.