Design Overview

Team: W's Only

- **Kevin Wang** *(kwang43)*
- **Malcolm Neill** *(mneill)*
- **Nicolette Miller** *(nmiller2)*
- **Edmund Yu** *(eyu9)*

Bon Chance Paging

This madman's paging algorithm has these modifications:

1. Instead of putting inactive and invalid pages on the front of the free list, it will look at the page number if it is even put them on the front, and if it is odd then put them on the rear.
2. Instead of starting the activity count at 5, it will assign the activity count a random value from 1–32.
3. Instead of subtracting from the activity count, it will divide the activity count by two, subtract one, and move it to the front of the active list instead of to the rear.
4. Finally, when it moves a page to the inactive list, it will choose a random number r from 1–10, and if r > 7 the page goes on the front instead of the rear.

With its arbitrary and convoluted paging decisions, 'good luck' with using it.

Design

Pageout Period

In order to see the effects of bon chance paging algorithm, we need to update the pageout period to 10 seconds from 600 seconds.

Inactive/Invalid to Free

We will use the value PAGE_SIZE to get the length of the page number in the physical address.

- If PAGE_SIZE = 4096, the page number is 20 bits long.
- If PAGE_SIZE = 8192, the page number is 19 bits long.
- If PAGE_SIZE = 16384, the page number is 18 bits long.
- If PAGE_SIZE = 32768, the page number is 17 bits long.

By using bitwise right shift, >> , we can parse the page number from `vm->phys_addr`. Finally we check if the resulting page number is even or odd using mod.

Activity Count Start

To change the starting activity count, simply update the define macro `ACT_INIT` from `5` to the expression `random() % 32` to start with a random number from 1 to 32. We will use `srandom(time_second)` to switch up the seed.

Since this parameter should be tunable, we need to add a modifier which we can use to add a bottom limit to the randomly generated activity count. We will use `sysctl(9)` to change the modifier (default 0) while the system is running.

Activity Count Update

When updating the activity count during pageout, we simply remove the line that subtracts from the activity count and replace it with: `m->act_count = m->act_count/2 - 1`

Active to Inactive

To generate a random number from 1 to 10, we use `random() % 10`. We will use `srandom(time_second)` to switch up the seed. Given an active page, we will enqueue into the tail (default FreeBSD) if the random number is less than or equal to 7, otherwise we will piggyback off `vm_page_deactivate_noreuse` which will bypass the LRU and move the specified page to the head of the inactive queue.

Statistics

We will need to add in global variables to track statistics and add a print statement that prints these stats at the end of the while(TRUE) loop in `vm_pageout_worker`.

We will not use log() as it is easier to just call the command `$dmesg | grep assgn3`.

Paging Modifications

Data

vm_pageout.c

```
vm_pageout_init(...) {
  ...
  vm_pageout_update_period = 10;  // change pageout period to every 10 seconds
  ...
}
```

vm_page.h

```
#define ACT_INIT random() % 32    // set initial activity count to random number
from 1 to 32
```

vm_page.c

```
static int act_count_min = 0;     // tuneable parameter used to set minimum
activity count
static int pos_modifier = 0;      // tuneable parameter used to set minimum for
inserting at HEAD or TAIL
```

Functions

vm_pageout.c

```
vm_pageout_scan_active(...) {
  ...
  act_scan:
    ...
    act_count = act_count/2 + 1;  // new way of updating the activity count
    ...
  ...
}
```

vm_page.c

```
vm_page_activate(...) {
  ...
  act_count =                     // allows for parameter to be tuneable
    ACT_INIT + act_count_min;     // wherever act_count is initialized
  ...
}
```

```
vm_page_deactivate(...) {
  seed random
  initialize random pos_select
  pos_select += pos_modifier      // allows for parameter to be tuneable
  ...
  if (page is not already inactive) {
    ...
    if (pos_select <= 7) {
      enqueue to tail
    } else {
      enqueue to head
    }
  } ...
}
```

vm_phys.c

```
vm_freelist_add (...) {
  int page_num;                   // buffer for getting the page number of a page
  ...
  if (PAGE_SIZE == 4096) {        // get page number from physical address
    get vm->phys_addr
    shift bits right by 12
  } elif (PAGE_SIZE == 8192) {
    '' by 13
  } elif (PAGE_SIZE == 16384) {
    '' by 14
  } elif (PAGE_SIZE == 32768) {
    '' by 15
  } else {default FreeBSD code}

  if (page_num is even)           // if page number is even, insert at front
    TAILQ_INSERT_HEAD;
  else                            // if page number is odd, insert at back
    TAILQ_INSERT_TAIL;
}
```

Benchmark Modifications

vm_page.c

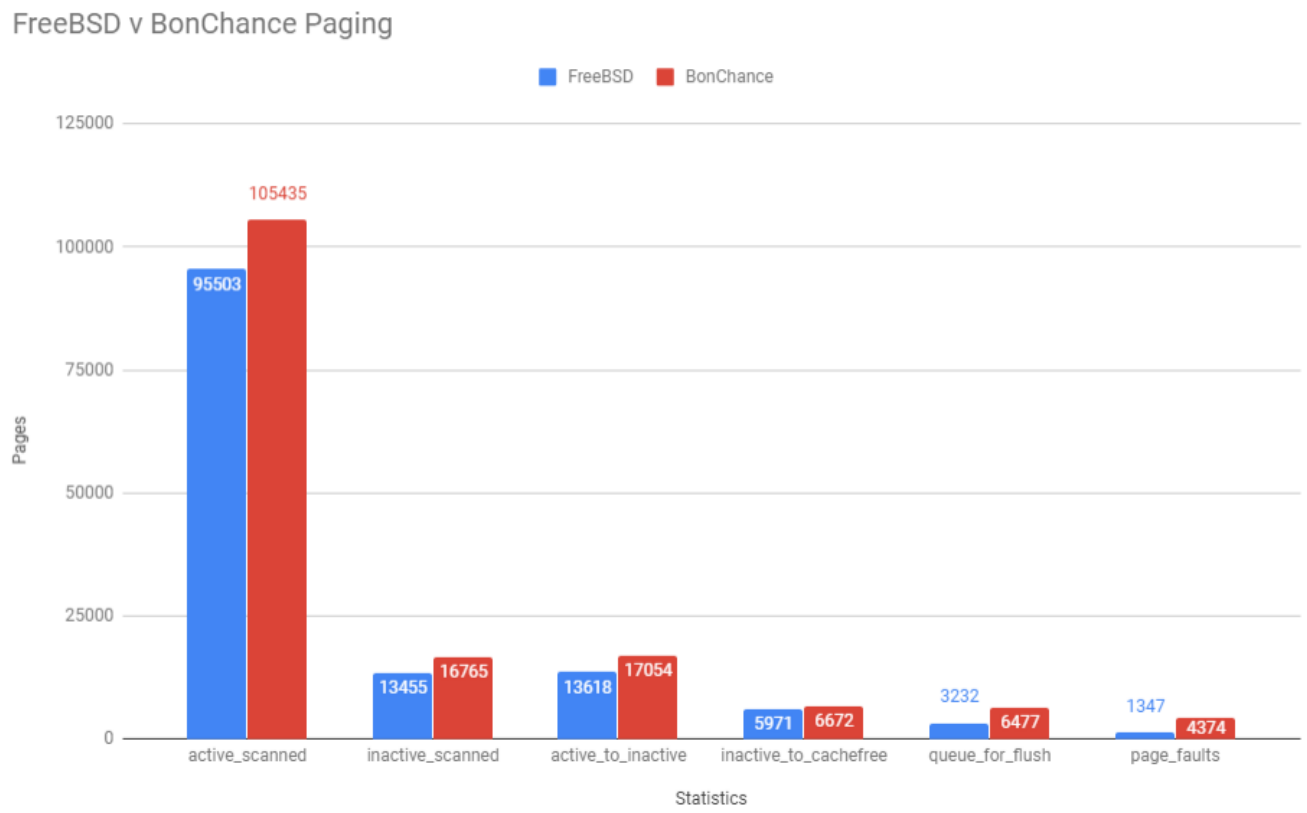
```
static int active_to_inactive     // counts # of pages move from active queue to
inactive queue
```

vm_pageout.c

```
static int active_scanned         // counts # of active pages scanned
static int inactive_scanned       // counts # of inactive pages scanned
static int queue_flush            // counts # of pages queued for flush
static int inactive_to_cachefree  // counts # of pages from inactive queue to
cache/free
```

```
vm_pageout_worker(...) {
  ...
  print all stats w/assgn3 tag
  reset all stats to 0
  ...
}
```

Analysis



The results showed an increase in page faults which is to be expected because Bon Chance is not optimal due to its random insertion into the queue. If the page is used often, it may still be shuffled -- resulting in a page fault.

Note that the activity count decay takes about the same number of passes, given 32 is (2 power 5). However, as the pages are then moved to the front of the active list, they constantly decay so pages are deactivated faster than usual.

We expected around the same number of pages to be moved to the front as there are moved to the back due to the quantity of odd/even page numbers being similar. However, we expected that more pages would be moved to the rear of the inactive list due to 1 to 7 being a larger range. This is also why there are more pages in the inactive queue than usual.

Overall, with Bon Chance, there is generally a lot more moving around, mostly due to the randomness of positioning the page within queues.

Sources

- "Design and Implementation of the FreeBSD Operating Systems"
- Piazza
- http://www.leidinger.net/FreeBSD/dox/kern/html/
- https://www.freebsd.org/doc/en/books/handbook/kernelconfig.html
- https://www.freebsd.org/doc/handbook/kernelconfig/building.html