Week8-Assignment-Report

11911609 葛兆宁

Q1

- When the OS find a page not in physical memory, it will issues a page fault, the the page fault handler will find the page's physical address in disk(or other hardware) by the page's PTE. Then, the OS will load the page into memory, change its physical address and relative information (include the present bit) in PTE. The TLB will be updated too(it'll be seen as a miss). If the memory is full, the OS will swap out some pages into the swap space (generally in disk or other hardware) first to support enough space for page in disk to load in.

Q2

- FIFO: 10
- LRU: 8
- MIN: 8

Q3

- code(Q3)

```
list_entry_t pra_list_head, *curr_ptr;
static int
_clock_init_mm(struct mm_struct *mm)
{
    // TODO
    list_init(&pra_list_head);
    mm->sm_priv = &pra_list_head;
    curr_ptr = &pra_list_head;
    return 0;
}

static int
_clock_map_swappable
(struct mm_struct *mm, uintptr_t addr, struct Page *page, int swap_in)
{
    // TODO
    // cprintf("list length: %d, list space: %d", cnt, mm->map_count);
    list_entry_t *head = (list_entry_t *)mm->sm_priv;
    list_entry_t *entry = &(page->pra_page_link);
    pte_t *pte = get_pte(mm->pgdir, page->pra_vaddr, 0);
    *pte |= PTE_A;
    assert(entry != NULL && head != NULL);
    // record the page access situlation
    list_add(curr_ptr, entry);
    curr_ptr = entry;
    return 0;
}

static int
_clock_swap_out_victim
(struct mm_struct *mm, struct Page **ptr_page, int in_tick)
{
    // TODO
    list_entry_t *head = (list_entry_t *)mm->sm_priv;
    assert(head != NULL);
    assert(in_tick == 0);
    curr_ptr = list_next(curr_ptr);
    list_entry_t *entry = curr_ptr;
    int flag = 0;

    if (list_next(head) == head)
    {
        *ptr_page = NULL;
        return 0;
    }

    while (true)
    {

        if (entry == head)
        {
            entry = list_next(entry);
            continue;
        }

        *ptr_page = le2page(entry, pra_page_link);
        pte_t *pte = get_pte(mm->pgdir, (*ptr_page)->pra_vaddr, 0);
        bool accessed = *pte & PTE_A;

        if (accessed == false)
        {
            curr_ptr = list_prev(entry);
            list_del(entry);
            flag = 1;
            break;
        }
        else
        {
            *pte &= ~(PTE_A);
            entry = list_next(entry);
        }
    }
    return 0;
}
```

- result(Q3)

```
11911609JohnnyGe@johnny-Ge-WXX9:~/OS/assignments/week8/week8_exe$ make qemu
+ cc kern/mm/swap_clock.c
+ ld bin/kernel
riscv64-unknown-elf-objcopy bin/kernel --strip-all -O binary bin/ucore.bin

OpenSBI v0.6
   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|

Platform Name          : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs     : 8
Current Hart           : 0
Firmware Base          : 0x80000000
Firmware Size          : 120 KB
Runtime SBI Version    : 0.2

MIDELEG : 0x0000000000000222
MEDELEG : 0x000000000000b109
PMP0    : 0x0000000080000000-0x000000008001ffff (A)
PMP1    : 0x0000000000000000-0xffffffffffffffff (A,R,W,X)
 os is loading ...

memory management: default_pmm_manager
membegin 80200000 memend 88000000 mem_size 7e00000
physcial memory map:
  memory: 0x07e00000, [0x80200000, 0x87ffffff].
check_alloc_page() succeeded!
check_pgdir() succeeded!
check_boot_pgdir() succeeded!
check_vma_struct() succeeded!
Store/AMO page fault
page falut at 0x00000100: K/W
check_pgfault() succeeded!
check_vmm() succeeded.
SWAP: manager = clock swap manager
BEGIN check_swap: count 3, total 31660
setup Page Table for vaddr 0X1000, so alloc a page
setup Page Table vaddr 0~4MB OVER!
set up init env for check_swap begin!
Store/AMO page fault
page falut at 0x00001000: K/W
Store/AMO page fault
page falut at 0x00002000: K/W
```

```
Store/AMO page fault
page falut at 0x00003000: K/W
Store/AMO page fault
page falut at 0x00004000: K/W
set up init env for check_swap over!
---------Clock check begin----------
write Virt Page c in clock_check_swap
write Virt Page a in clock_check_swap
write Virt Page d in clock_check_swap
write Virt Page b in clock_check_swap
write Virt Page e in clock_check_swap
Store/AMO page fault
page falut at 0x00005000: K/W
swap_out: i 0, store page in vaddr 0x1000 to disk swap entry 2
write Virt Page b in clock_check_swap
write Virt Page a in clock_check_swap
Store/AMO page fault
page falut at 0x00001000: K/W
swap_out: i 0, store page in vaddr 0x3000 to disk swap entry 4
swap_in: load disk swap entry 2 with swap_page in vadr 0x1000
write Virt Page b in clock_check_swap
write Virt Page c in clock_check_swap
Store/AMO page fault
page falut at 0x00003000: K/W
swap_out: i 0, store page in vaddr 0x4000 to disk swap entry 5
swap_in: load disk swap entry 4 with swap_page in vadr 0x3000
write Virt Page d in clock_check_swap
Store/AMO page fault
page falut at 0x00004000: K/W
swap_out: i 0, store page in vaddr 0x5000 to disk swap entry 6
swap_in: load disk swap entry 5 with swap_page in vadr 0x4000
write Virt Page e in clock_check_swap
Store/AMO page fault
page falut at 0x00005000: K/W
swap_out: i 0, store page in vaddr 0x2000 to disk swap entry 3
swap_in: load disk swap entry 6 with swap_page in vadr 0x5000
write Virt Page a in clock_check_swap
Clock check succeed!
check_swap() succeeded!
```

**(mainly because of the lack of the long screenshot in ubuntu System,and I don't want to joint two seperate screenshots that will give a bad feeling in checking. So, I copy the the display in the terminal and paste in here)*

Q4

- code(Q4)

```
list_entry_t pra_list_head, *curr_ptr;

static int
_lru_init_mm(struct mm_struct *mm)
```

```
{
    // TODO
    list_init(&pra_list_head);
    curr_ptr = &pra_list_head;
    mm->sm_priv = &pra_list_head;
    return 0;
}

static int
_lru_map_swappable
(struct mm_struct *mm, uintptr_t addr, struct Page *page, int swap_in)
{
    // TODO

    list_entry_t *head = (list_entry_t *)mm->sm_priv;
    list_entry_t *entry = &(page->pra_page_link);

    assert(entry != NULL && head != NULL);
    page->pra_vaddr = addr;

    *(unsigned char *)page->pra_vaddr++;

    list_add(head, entry);

    return 0;
}

static int
_lru_swap_out_victim
(struct mm_struct *mm, struct Page **ptr_page, int in_tick)
{
    // TODO
    list_entry_t *head = (list_entry_t *)mm->sm_priv;
    assert(head != NULL);
    assert(in_tick == 0);
    int min = 100000;
    if (list_next(head) == head)
    {
        *ptr_page = NULL;
        return 0;
    }
    curr_ptr = list_next(head);
    list_entry_t *final_entry = NULL;
    while (curr_ptr != head)
    {
        int tmp =
        *(unsigned char *)(le2page(curr_ptr, pra_page_link)->pra_vaddr);
        if (tmp <= min)
        {
            min = tmp;
            final_entry = curr_ptr;
        }
        curr_ptr = list_next(curr_ptr);
    }
    if (final_entry != NULL)
    {
        curr_ptr = list_prev(final_entry);
        list_del(final_entry);
        *ptr_page = le2page(final_entry, pra_page_link);
    }
    else
    {
        *ptr_page = NULL;
    }
    return 0;
}
```

- result(Q4)

```
11911609JohnnyGe@johnny-Ge-WXX9:~/OS/assignments/week8/week8_exe$ make qemu
+ cc kern/init/entry.S
+ cc kern/init/init.c
+ cc kern/libs/stdio.c
+ cc kern/debug/panic.c
+ cc kern/debug/kdebug.c
+ cc kern/debug/kmonitor.c
+ cc kern/driver/ide.c
+ cc kern/driver/clock.c
+ cc kern/driver/console.c
+ cc kern/driver/intr.c
+ cc kern/trap/trap.c
+ cc kern/trap/trapentry.S
+ cc kern/mm/swap_lru.c
+ cc kern/mm/vmm.c
+ cc kern/mm/swap.c
+ cc kern/mm/swap_fifo.c
+ cc kern/mm/default_pmm.c
+ cc kern/mm/swap_clock.c
+ cc kern/mm/pmm.c
+ cc kern/fs/swapfs.c
+ cc libs/string.c
+ cc libs/printfmt.c
+ cc libs/readline.c
+ cc libs/rand.c
+ ld bin/kernel
riscv64-unknown-elf-objcopy bin/kernel --strip-all -O binary bin/ucore.bin

OpenSBI v0.6
   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
         | |
         |_|

Platform Name          : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs     : 8
Current Hart           : 0
Firmware Base          : 0x80000000
Firmware Size          : 120 KB
Runtime SBI Version    : 0.2

MIDELEG : 0x0000000000000222
MEDELEG : 0x000000000000b109
PMP0    : 0x0000000080000000-0x000000008001ffff (A)
PMP1    : 0x0000000000000000-0xffffffffffffffff (A,R,W,X)
 os is loading ...

memory management: default_pmm_manager
membegin 80200000 memend 88000000 mem_size 7e00000
physcial memory map:
  memory: 0x07e00000, [0x80200000, 0x87ffffff].
check_alloc_page() succeeded!
check_pgdir() succeeded!
check_boot_pgdir() succeeded!
check_vma_struct() succeeded!
Store/AMO page fault
page falut at 0x00000100: K/W
check_pgfault() succeeded!
check_vmm() succeeded.
SWAP: manager = lru swap manager
BEGIN check_swap: count 3, total 31660
setup Page Table for vaddr 0X1000, so alloc a page
setup Page Table vaddr 0~4MB OVER!
set up init env for check_swap begin!
Store/AMO page fault
page falut at 0x00001000: K/W
Store/AMO page fault
page falut at 0x00002000: K/W
Store/AMO page fault
page falut at 0x00003000: K/W
Store/AMO page fault
page falut at 0x00004000: K/W
set up init env for check_swap over!
---------LRU check begin----------
write Virt Page 3 in lru_check_swap
write Virt Page 1 in lru_check_swap
write Virt Page 4 in lru_check_swap
write Virt Page 2 in lru_check_swap
write Virt Page 5 in lru_check_swap
Store/AMO page fault
page falut at 0x00005000: K/W
swap_out: i 0, store page in vaddr 0x3000 to disk swap entry 4
write Virt Page 3 in lru_check_swap
Store/AMO page fault
```

```
page falut at 0x00003000: K/W
swap_out: i 0, store page in vaddr 0x1000 to disk swap entry 2
swap_in: load disk swap entry 4 with swap_page in vadr 0x3000
write Virt Page 1 in lru_check_swap
Store/AMO page fault
page falut at 0x00001000: K/W
swap_out: i 0, store page in vaddr 0x4000 to disk swap entry 5
swap_in: load disk swap entry 2 with swap_page in vadr 0x1000
write Virt Page 4 in lru_check_swap
Store/AMO page fault
page falut at 0x00004000: K/W
swap_out: i 0, store page in vaddr 0x2000 to disk swap entry 3
swap_in: load disk swap entry 5 with swap_page in vadr 0x4000
write Virt Page 4 in lru_check_swap
write Virt Page 5 in lru_check_swap
write Virt Page 2 in lru_check_swap
Store/AMO page fault
page falut at 0x00002000: K/W
swap_out: i 0, store page in vaddr 0x3000 to disk swap entry 4
swap_in: load disk swap entry 3 with swap_page in vadr 0x2000
write Virt Page 3 in lru_check_swap
Store/AMO page fault
page falut at 0x00003000: K/W
swap_out: i 0, store page in vaddr 0x1000 to disk swap entry 2
swap_in: load disk swap entry 4 with swap_page in vadr 0x3000
LRU check succeed!
check_swap() succeeded!
```