HW4 – Virtual Memory

Speaker: Wei-Xiang Wang

M10715010@mail.ntust.edu.tw

Main Memory in Nachos

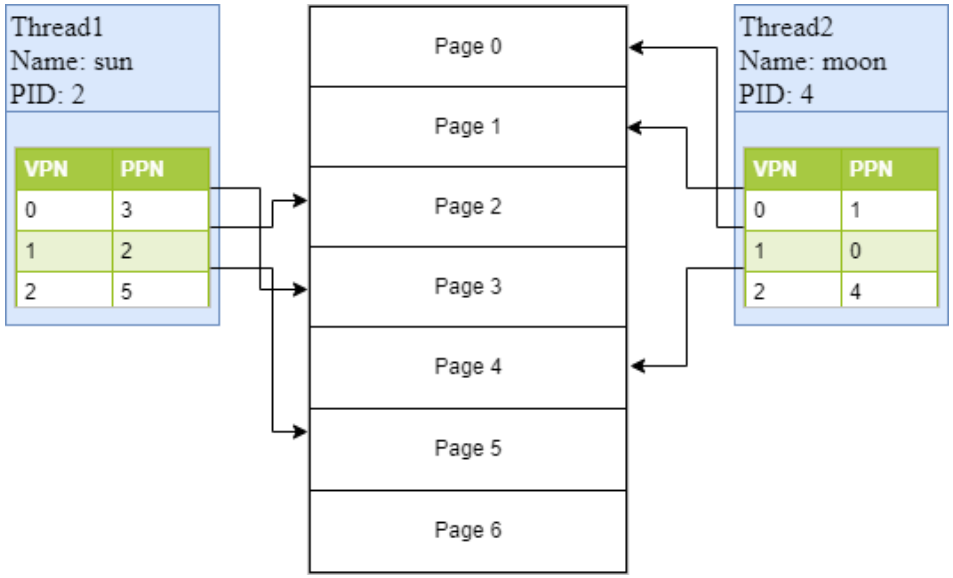
- Main memory is implemented as an array of bytes
- By default, a frame has 128 bytes in Nachos
  - Thus, Nachos has 128 bytes * 32 frames = 4KB of Main Memory
- See machine/machine.h
  - const unsigned int PageSize = 128;
  - const unsigned int NumPhysPages = 32;
  - const int MemorySize = (NumPhysPages * PageSize);

Memory Management Unit

- Machine 利用 TranslationEntry 創建的 PageTable 去處理正在運行的 process。
- 如同上一次作業的 virtual memory 問題，每個 thread 都擁有自己的 logical address space，並且映射到 Nachos 的 physical address space。
  - 所以每個 thread 都擁有自己的 AddrSpace 成員
    - threads/thread.h
      - AddrSpace *space; // User code this thread is running.

Memory Management Unit (Cont.)

- 我們可以存取 thread 的頁表
  - currentThread->space->pageTable
- VPN: Virtual Page Number　　PPN: Physical Page Number



Loading Program Flow

1. 透過 NachOS 執行 Program(“TEST”)
2. 創建 Thread(“TEST”) 和它的 page table
3. 讀取 Program(“TEST”) 到 main memory
   - AddrSpace::Load() in userprog/addrspace.cc
4. 將 Thread(“TEST”) 放到 readyList 等待
5. kernel->machine->Run()
6. 重複執行程式直到拋出 exit 的 exception
   - If page fault is true then load the demanded page

Translation Entries

- machine/translate.h

```
class TranslationEntry {
  public:
    unsigned int virtualPage;   // The page number in virtual memory.
    unsigned int physicalPage;  // The page number in real memory (relative to the
                        //  start of "mainMemory"
    bool valid;         // If this bit is set, the translation is ignored.
                        // (In other words, the entry hasn't been initialized.)
    bool readOnly;      // If this bit is set, the user program is not allowed
                        // to modify the contents of the page.
    bool use;           // This bit is set by the hardware every time the
                        // page is referenced or modified.
    bool dirty;         // This bit is set by the hardware every time the
                        // page is modified.
};
```

- userprog/addrspace.h
  - TranslationEntry *pageTable;

Translate

- 當 process 使用 logical memory address 時，透過 NachOS 的 Translate() 去獲得對應的 physical address，如果 page 的 valid 是 true 的話，則做對應的轉換即可。
- 如果 page 的 valid 是 false，則 NachOS 會拋出 page fault exception，並且由 exception handler 去做替換頁表的動作，然後再重新執行一次剛剛的指令。
  1. Translate.cc (Machine::ReadMem)
  2. RaiseException
  3. ExceptionHandler
  4. Swap the page into main memory
  5. Restart the instruction

Homework Requirements

- 實現 virtual memory 並可以處理 page fault
  - 驗證結果請使用測試程式，宣告足夠大的 int array，並對 array進行一些操作 (e.g. 累加array的值並輸出)
  - 最好可以附上在實作 virtual memory 之前，運行這個測試程式會因為 main memory 不夠大而造成 segment fault 的截圖
- 實作 Page Replacement Algorithm 的 FIFO 與 LRU
  - First in First out Page Replacement(FIFO)
  - Least Recently Used Page Replacement(LRU)
  - 需在報告內告訴該如何切換演算法
  - 需於截圖上顯示出可識別不同演算法的結果

Tip1

- Trace the following files
  - code/machine/machine.cc
  - code/machine/translate.cc
  - code/machine/interrupt.cc
  - code/machine/interrupt.cc
  - code/userprog/addrspace.cc
  - code/userprog/exception.cc
  - code/threads/main.cc
  - code/threads/thread.cc
  - code/threads/scheduler.cc
  - All the above header file (.h)

Tip2

- 實現輔助記憶體的方法不限。
  - (e.g. SynchDisk, Thread structure..)
- 可透過 exception handler 方式處理 page faults
  - 必須自己加上 PageFaultException case 去處理
  - 處理 page faults 時不要增加 program counter
- 可透過 Machine::Translate 直接替換 page
  - 修改某些成員的結構，增加額外的記憶體頁表作輔助

Hand in source code & report

- Source Code
  - Push it to your GitHub
- Report
  - The work distribution of team members
  - The idea you take to implement the problem in NachOS
  - Some important codes and comments
  - Experiment result (Screenshot)
  - Saved as [Team ID]_HW4_report.pdf
    - e.g. 2_HW4_report.pdf
  - Upload the report to moodle